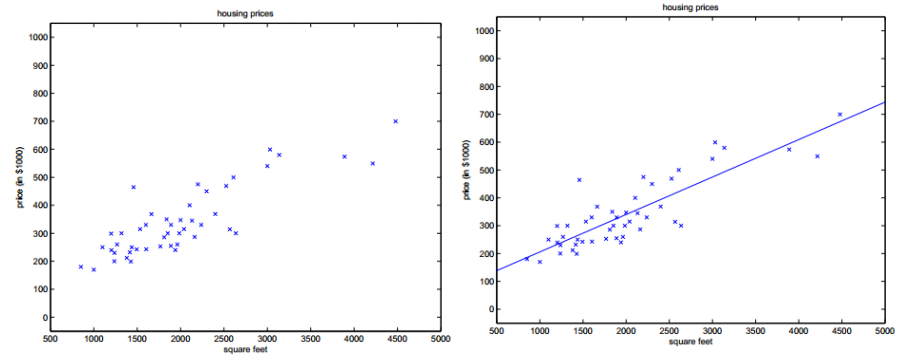
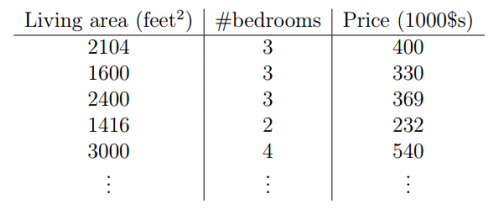
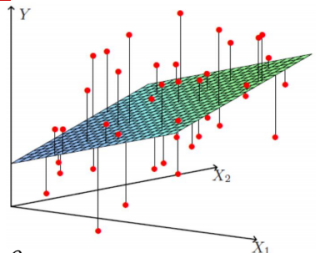
# 线性回归模型

## 1.1房价问题

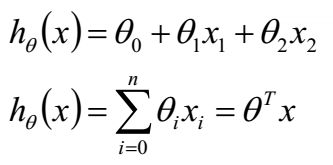
考虑房屋的面积和房屋的售价，这里我们可以用y=kx+b预测



模拟数据如下：

建立房屋面积、卧室数目和售价的模型（思考：如何使得模型最优？）：



当有了新的房屋面积数据即可预测房价。

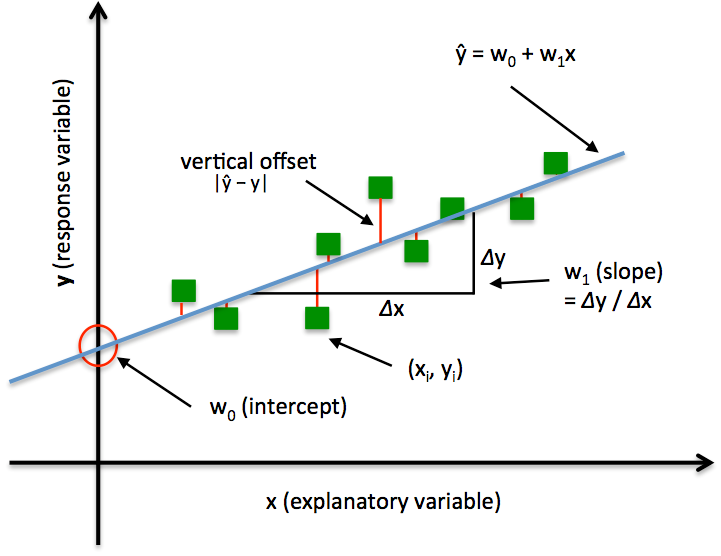
## 1.2回归模型

回归模型（regression model）用于连续型变量的预测分析。

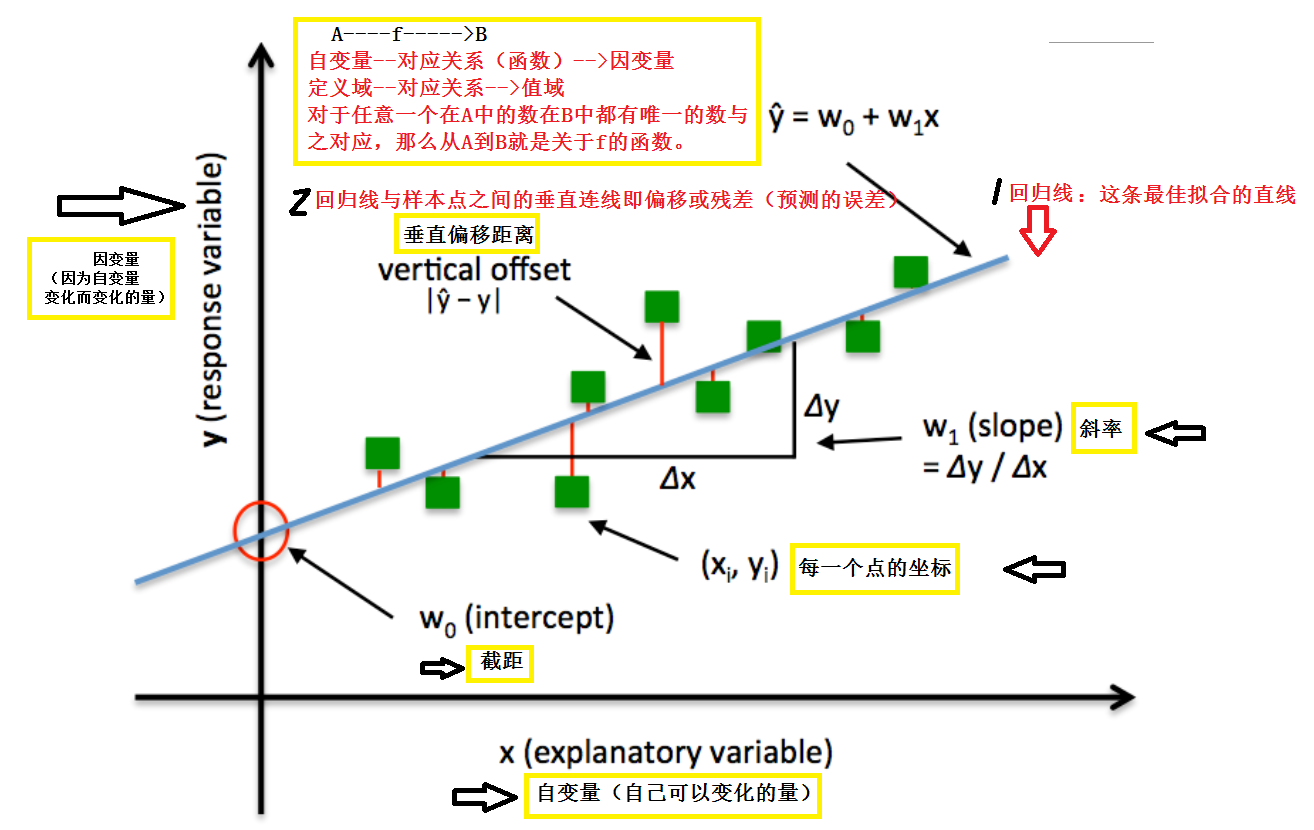
通常在只有一个解析变量的时候，线性回归称为简单线性回归，如y=kx+b；当有多个解析变量的时候，称之为多元线性回归，，其中w0是x0=1时在y轴上的截距。



线性回归模型的目标：通过学习得到线性方程的这两个权值，如在y=kx+b中，得到k和b两个权值，并用这个方程解释变量和目标变量之间的关系。



笔记：



# 一、简单线性回归

## 1. 前提介绍：

为什么需要统计量？

统计量：描述数据特征

### 1.1集中趋势衡量

（1）均值（平均数，平均值）（mean）

数学表达式：

举例：{5, 3, 8, 2, 2}

(5 + 3 + 8 + 2+ 2) / 5 = 20 / 5 = 4

（2）中位数 （median）: 将数据中的各个数值按照大小顺序排列，居于中间位置的变量

举例： 给数据排序：1， 2， 2， 6， 9

中位数即为（找出位置处于中间的变量）：2

规律： 当n为基数的时候：直接取位置处于中间的变量

当n为偶数的时候，取中间两个量的平均值

（3）众数 （mode）：数据中出现次数最多的数

举例： 给数据序列：1， 2， 2， 6， 9

众数为：2

1. 四分位数：四分位数（Quartile）把所有数值由小到大排列并分成四等份，处于三个分割点位置的数值就是四分位数。QL=下四分位数，即第25百分位数（n / 4）；QU=上四分位数，即第75百分位数（3n/ 4）。

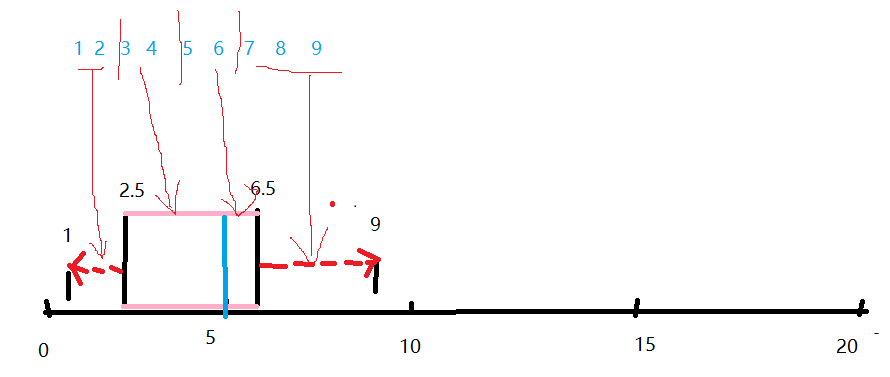
举例1：原序列：1，3，5，7，8，9，2，4，6

经过排序后的数字 1 2 3 4 5 6 7 8 9

找到中间的数5，左边是1 2 3 4，右边是6 7 8 9 （5就是中位数）

左边1 2 3 4的中位数为(2+3)/2=2.5 (2.5是下四分位数)

右边 6 7 8 9的中位数为(7+8)/2=7.5 (7.5是上四分位数)



举例2：以序列(1,2,3,4)为例，分析更一般的求解四分位数的方法

　首先确定四分位数的位置，在根据位置寻找数值：

　　Q1的位置=（n+1）/4 Q1数值=第一个数\*0.25+第二个数\*0.75

　　Q2的位置=（n+1）/2　 Q2数值=第一个数\*0.5+第二个数\*0.5

　　Q3的位置=3（n+1）/4 Q3数值=第一个数\*0.75+第二个数\*0.25

Q1位置=(4+1)/4=1.25

Q1对应的数值为1\*0.25+2\*0.75=1.75

Q2位置=(4+1)/2=2.5

Q2对应的数值为2\*0.5+3\*0.5=2.5

Q3位置=3\*（4+1）/4=3.75

Q3对应位置数值为3\*0.75+4\*0.25=3.25

### 1.2.离散程度衡量

（1）方差（variance)

数学公式：

举例：给定数据集{6, 2, 9, 1, 2}

a. (6 - 4)^2 + (2 - 4) ^2 + (9 - 4)^2 + (1 - 4)^2 + (2 - 4)^2

   = 4 + 4 + 25 + 9 + 4

   = 46

b. n - 1 = 5 - 1 = 4

c. 46 / 4 = 11.5

（2）标准差 (standard deviation)

数学公式：

s = sqrt(11.5) = 3.39

1. 四分位差：四分位差(quartile deviation)，也称为内距或四分间距(inter-quartile range)，它是上四分位数(QU，即位于75%)与下四分位数(QL，即位于25%)的差。

举例：以上面的序列：1，3，5，7，8，9，2，4，6为例，上四分位数为2.5，下四分位数为7.5，则四分位差是7.5-2.5=5

1. 极差：全距(Range)，又称极差，是用来表示统计资料中的变异量数(measures of variation)，其最大值与最小值之间的差距

举例：以上面的序列：1，3，5，7，8，9，2，4，6为例，最大值为9，最小值为1，则极差为8。

（5）离散系数：离散系数又称变异系数，CV(Coefficient of Variance)表示。CV(Coefficient of Variance)：标准差与均值的比值。离散系数越小，数据的离散程度就越小，反之，亦然。

### 1.3偏度与峰度的度量

（1）偏态系数：偏度(Skewness)亦称偏态、偏态系数，偏度是统计数据分布偏斜方向和程度的度量，是统计数据分布**非对称程度的数字特征**。Sk>0时，分布呈正偏态（右偏），Sk<0时，分布呈负偏态（左偏）。

（2）峰态系数：（Kurtosis)峰度系数是用来反映频数分布曲线顶端尖峭或扁平程度的指标。在正态分布情况下，峰度系数值是3。>3的峰度系数说明观察量更集中，有比正态分布更短的尾部；<3的峰度系数说明观测量不那么集中，有比正态分布更长的尾部，类似于矩形的均匀分布。峰度系数的标准误用来判断分布的正态性。峰度系数与其标准误的比值用来检验正态性。如果该比值绝对值大于2，将拒绝正态性。

### 1.3 Python代码实战

Python2代码

#-\*-coding:utf8-\*-  
import numpy as np  
import scipy.stats as sts  
scores=[31,24,23,25,14,25]  
#1.集中程度的度量  
print '求和：',np.sum(scores)  
print '平均数：',np.mean(scores)  
print '中位数：',np.median(scores)  
print '众数',sts.mode(scores)#众数 ModeResult(mode=array([25]), count=array([2]))  
print '四分位数',sts.mstats.mquantiles(scores,prob=[0.25, 0.5, 0.75])#四分位数 [ 22.55 24.5 25.3 ]  
#2.离散程度的度量  
print '最大值：',np.max(scores)  
print '最小值',np.min(scores)  
print '极差',np.max(scores)-np.min(scores)  
print '标准差',np.std(scores)  
print '方差',np.var(scores)  
print '离散系数',np.std(scores)/np.mean(scores)  
# 最大值： 31 最小值 14 极差 17  
# 标准差 5.02217305777 方差 25.2222222222  
# 离散系数 0.212204495399  
#3.偏态和峰态  
import pandas as pd  
s1=pd.Series(scores)  
print type(s1)  
print '偏态',s1.skew()  
print '峰态',s1.kurt()  
# 偏态 -0.908840574263 峰态 2.63720429273

Python3代码

以下代码基于Python3.5环境编写

import numpy as np

import stats as sts

scores = [31, 24, 23, 25, 14, 25, 13, 12, 14, 23,

32, 34, 43, 41, 21, 23, 26, 26, 34, 42,

43, 25, 24, 23, 24, 44, 23, 14, 52,32,

42, 44, 35, 28, 17, 21, 32, 42, 12, 34]

#集中趋势的度量

print('求和：',np.sum(scores))

print('个数：',len(scores))

print('平均值:',np.mean(scores))

print('中位数:',np.median(scores))

print('众数:',sts.mode(scores))

print('上四分位数',sts.quantile(scores,p=0.25))

print('下四分位数',sts.quantile(scores,p=0.75))

#离散趋势的度量

print('最大值:',np.max(scores))

print('最小值:',np.min(scores))

print('极差:',np.max(scores)-np.min(scores))

print('四分位差',sts.quantile(scores,p=0.75)-sts.quantile(scores,p=0.25))

print('标准差:',np.std(scores))

print('方差:',np.var(scores))

print('离散系数:',np.std(scores)/np.mean(scores))

#偏度与峰度的度量

print('偏度:',sts.skewness(scores))

print('峰度:',sts.kurtosis(scores))

## 简单线性回归

### 0.引出

我们都参加过高考，据统计，高考的物理成绩确实与数学成绩有一定关系，但除此之外，还存在很多影响物理成绩的因素，例如：是否喜欢物理，用在物理上的时间等。而当我们主要考虑数学成绩对物理的影响时，就是要考察这两者之间的相关关系。

现实生活中还有很多的相关关系，如

1.商品销售输入与广告支出经费之间的关系，销售输入与广告支出有着密切的关系，但是还与商品质量、居民收入等因素有关。

2.粮食产量与施肥量之间的关系。在一定范围内，施肥量越大，粮食生产就越高。除此之外，粮食产量还受到土壤质量、降雨量等的影响。

3.人体内脂肪的含量与年龄之间的关系。在一定年龄段内，随着年龄的增长，人体内的脂肪含量会增加，但人体内的脂肪含量还和饮食习惯，体育锻炼有关系，可能还与先天

体质有关系。

对于上述两个变量之间的关系，应该说都可以根据经验做出相应的判断，因为“经验当中有规律”，但是，不管你经验多么丰富，如果只凭借经验办事，还是很容易出错。因此在分析两个变量之间的相关关系时，我们需要一些说服力的办法。

在寻找变量之间的相关关系中，统计同样发挥着非常重要的作用。因为上面提到的这种关系，并不像匀速直线运动中时间与速度的关系那样是完全确定的，而是带有不确定性，这就需要通过收集大量的数据（有时候通过调查、或实验），在对数据进行统计分析的基础上，发现其中的规律，才能让他们之间的关系做出判断。

**两个变量的线性相关：**

如下表中描述了人体的脂肪百分比和年龄的关系图表：

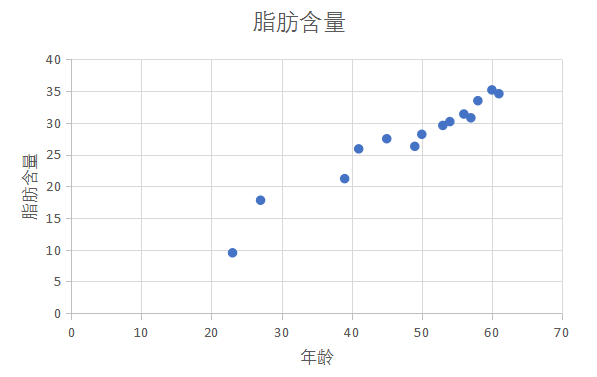
|  |  |
| --- | --- |
| 年龄 | 脂肪含量 |
| 23 | 9.5 |
| 27 | 17.8 |
| 39 | 21.2 |
| 41 | 25.9 |
| 45 | 27.5 |
| 49 | 26.3 |
| 50 | 28.2 |
| 53 | 29.6 |
| 54 | 30.2 |
| 56 | 31.4 |
| 57 | 30.8 |
| 58 | 33.5 |
| 60 | 35.2 |
| 61 | 34.6 |

**问题：**根据上述数据，人体的脂肪含量与年龄之间有怎么样的关系呢？

**对问题的描述：**

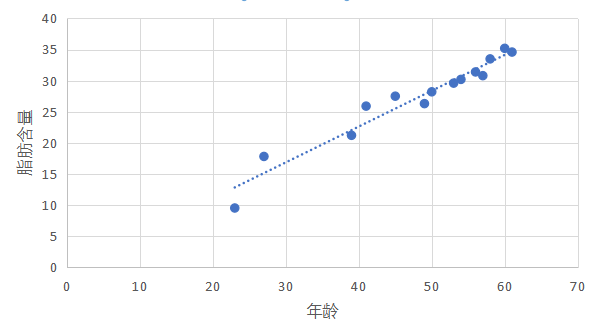
一般地，对于某个人来说，他的体内脂肪不一定随年龄增长而增加或减少，但是如果把很多个体放在一起，这时就可能表现出一定的规律性，各年龄对应的脂肪数据是这个年龄人群脂肪含量的样本平均值。观察上述表数据，从大体上看，随着年龄的增加，人体中脂肪的百分比也在增加。为了确定这一细节，我们需要进行数据的分析，与以前一样，我们可以做统计图、表，通过作统计图、表，可以使我们对两个变量之间的关系有一个直观的印象和判断。

下面我们做一个散点图，如图，假设人的年龄影响体内脂肪含量，于是，按照习惯，以x轴表示年龄，y轴表示脂肪含量，得到下图：



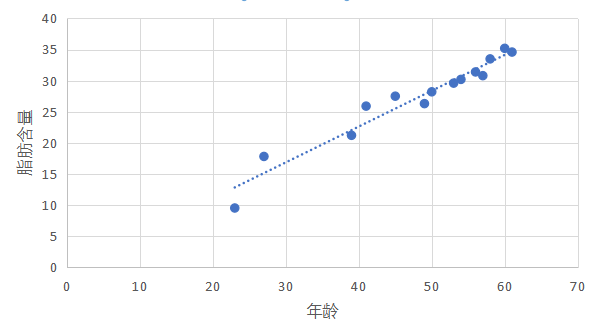
这些散分布的位置也是值得注意的，他们散布在从左上角到右下角的区域。对于两个变量的这种相关关系，我们称为正相关。还有一些变量，例如汽车的重量和汽车每消耗1L汽油所行驶的平均路程，是负相关，也就是汽车越重，每消耗1L汽油所行驶的平均路程就越短，这时的点如果绘制在画布上 将会从左上角到右下角的区域内。

接下来，需要进一步考虑的问题是，当人的年龄增加时，体内脂肪含量到底是以什么方式增加的呢？



从散点图可以看出，这些点大致分布在通过散点图中心的一条直线附近，如果散点图中的点分布从整体上看大致是在一条直线附近，我们就称这两个变量之间具有线性相关关系，这条直线叫做回归直线（regression line）。

如果能求出这条回归直线的方程，那么我们就可以清晰的了解年龄与体内脂肪含量的相关性，就像平均值可以作为一个变量的数据的代表一样，这条直线可以作为两个变量具有线性相关性的代表。



当你拿到这样一个任务的时候，你可能会采用测量的做法，先画出一条直线，测量各点与它的距离，然后移动直线，到达一个使得距离的和最小的位置，测量出此时的斜率和截距，既可以得到回归方程了。

也可能会采用平均方法，也就是在散点图中多取几组点，确定出几条直线的方程，在分别求出各条直线的斜率、截距的平均数，将这两个平均数作为回归方程的斜率和截距。

上面方法虽然有一定道理，但是总让人感觉可靠性不强。实际上，求回归方程的关键是如何用数学的方法来刻画“从整体上，各点与此直线的距离最小”。

假设我们已经得到两个具有线性相关关系的变量的一组数据



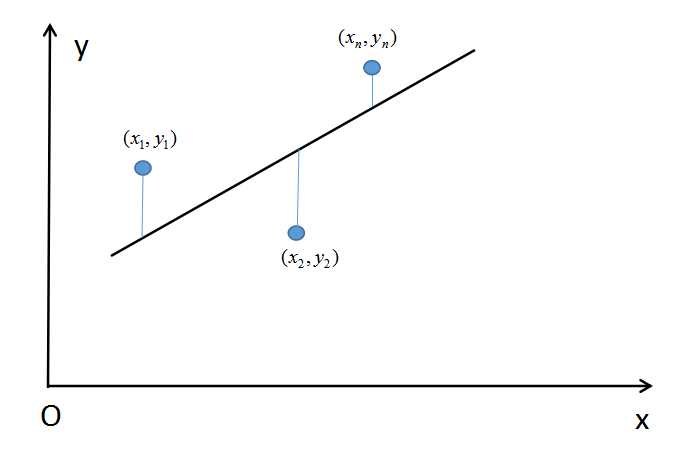
且所求的回归方程设为



它与实际数据的偏差是



在坐标轴中表示如下



这样，用这n个偏差的和来刻画“各点与此直线的整体偏差”比较合适的，由于可正可负，为了避免互相抵消，可以考虑用代替，但由于它含有绝对这，运算不方便，所以改用

即：



这样，问题就归结为：当a和b取什么值时Q最小，即总体偏差最小，经过数学上的最小值运算，a和b的值由下面式子给出



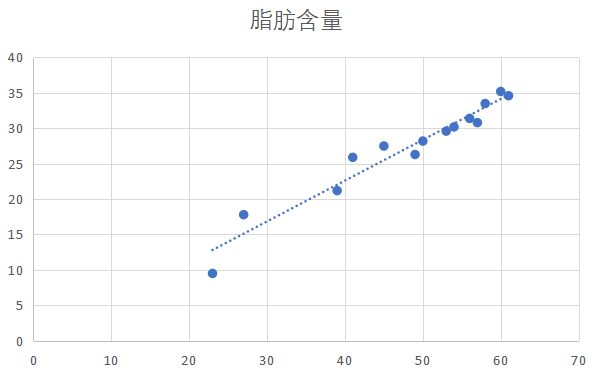


其中b是回归方程的斜率，a是截距。

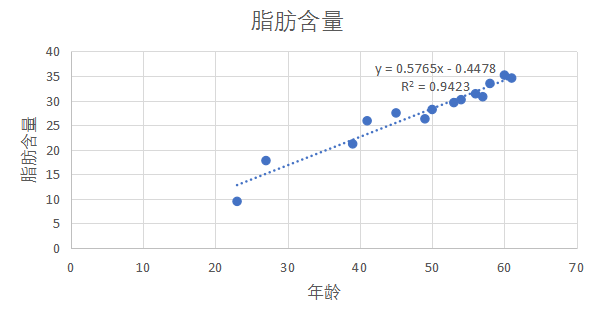
***总结：***这种通过求解Q关系式的最小值而得到回归直线的方法，即求**回归直线**，使得样本数据的点到它的距离的平方和最小的方法，叫做最小二乘法（Least square）。

***实操***：演示Execl画回归拟合直线：

1. 准备数据，首先绘制散点图
2. 在散点图基础上点”趋势预测”即可看到拟合的直线。



1. 在图示区域显示拟合直线的方程



### 回归和分类的区分：

回归：因变量Y是连续值（房价、人数、降雨量）

分类：因变量Y是离散值（颜色分类、电脑品牌、信誉）

区别：观察因变量Y的取值时候为连续的值。

### 2. 线性回归

1. 很多决策过程都是根据两个或多个变量之间的关系做决策
2. 回归分析：用来建立方程模拟两个或多个变量之间的关系。
3. 搞清楚：因变量Y和自变量X

### 3. 简单线性回归介绍(Simple Linear Regression)

1. 简单线性回归包含一个自变量和一个因变量
2. 以上两个变量勇一条直线模拟
3. 如果是两个以上的变量，就称之为多元回归分析

### 4. 简单线性回归模型

    （1）被用来描述因变量(y)和自变量(X)以及偏差(error)之间关系的方程叫做回归模型

（2） 简单线性回归的模型是:



  其中：参数，为偏差

### 5. 关于偏差ε的假定

     （1）是一个随机的变量，均值为0，E（ε）=0

     （2）ε的方差(variance)对于所有的自变量x是一样的

     （3）ε的值是独立的

     （4）ε满足正态分布

### 6.简单线性回归方程

    数学方程：E(y) = β0+β1x

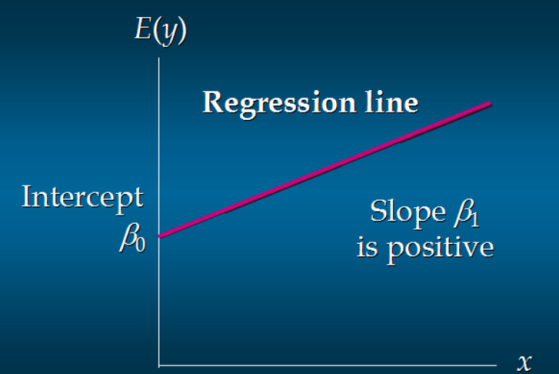
    这个方程对应的图像是一条直线，称作回归线

    其中，β0是回归线的截距

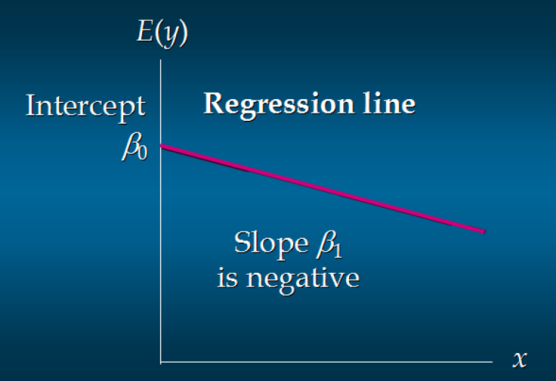
        β1是回归线的斜率

        E(y)是在一个给定x值下y的期望值（均值）

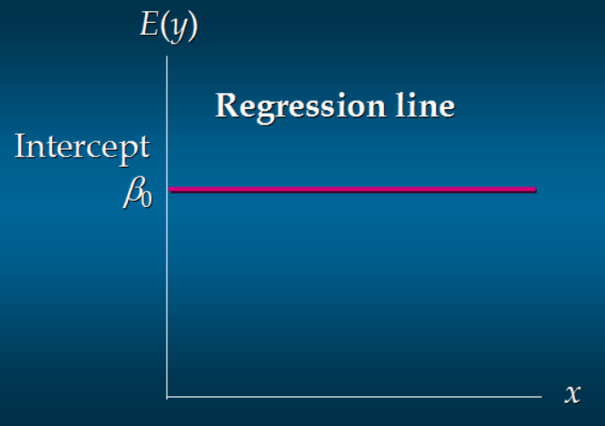
### 7.正向线性关系：



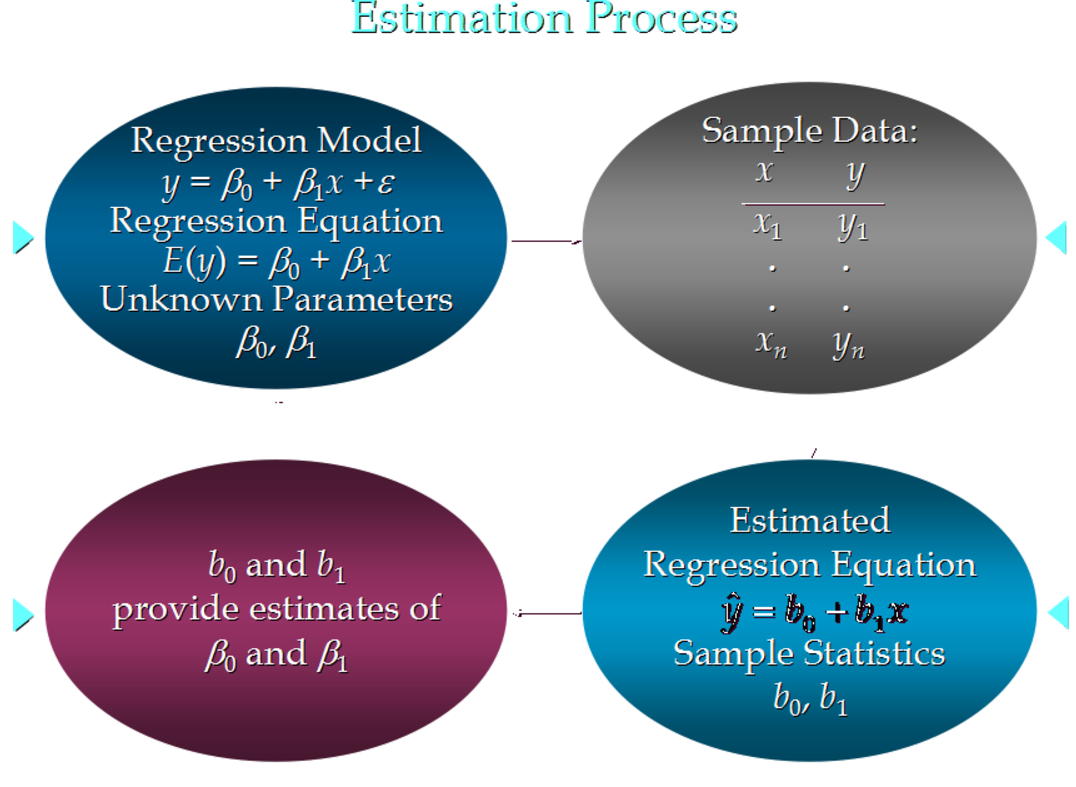
### 8.负向线性关系：

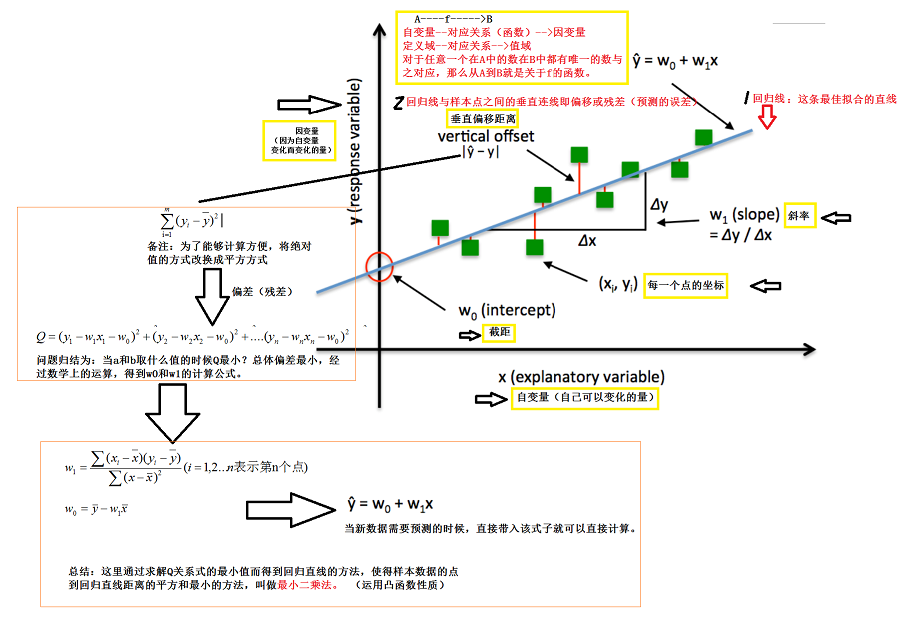


  8. 无关系



### 10.线性回归分析流程：





### 12.练习题

如下统计表为卖出的热饮杯数与当天气温的对比表：

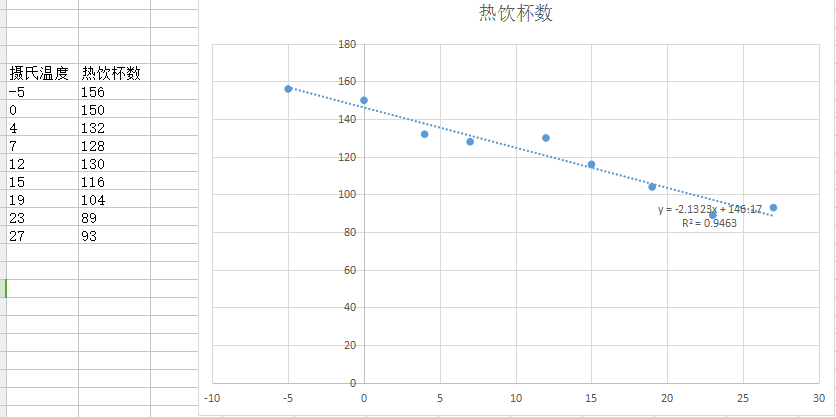
根据图表回答以下问题：

1. 画出散点图。
2. 从散点图中发现气温与卖出热饮杯数的一般规律。
3. 求回归方程。
4. 如果某天气温是2度，预测这天卖出的热饮数。

|  |  |
| --- | --- |
| 摄氏温度 | 热饮杯数 |
| -5 | 156 |
| 0 | 150 |
| 4 | 132 |
| 7 | 128 |
| 12 | 130 |
| 15 | 116 |
| 19 | 104 |
| 23 | 89 |
| 27 | 93 |
| 31 | 76 |
| 36 | 54 |

解答过程：

（1）拟合直线：



1. 从上图中可以看出各点散布在从左上角到右下角的区域，因此气温与热饮销量呈负相关，即气温越高，销售的热饮数量越少。
2. 看图中的直线方程
3. 当x=2时，，因此，某天的气温为2摄氏度，这天大概可以卖出143杯热饮。

## 3.简单线性回归模型举例及代码

### 1.汽车销售和投放广告销售预测

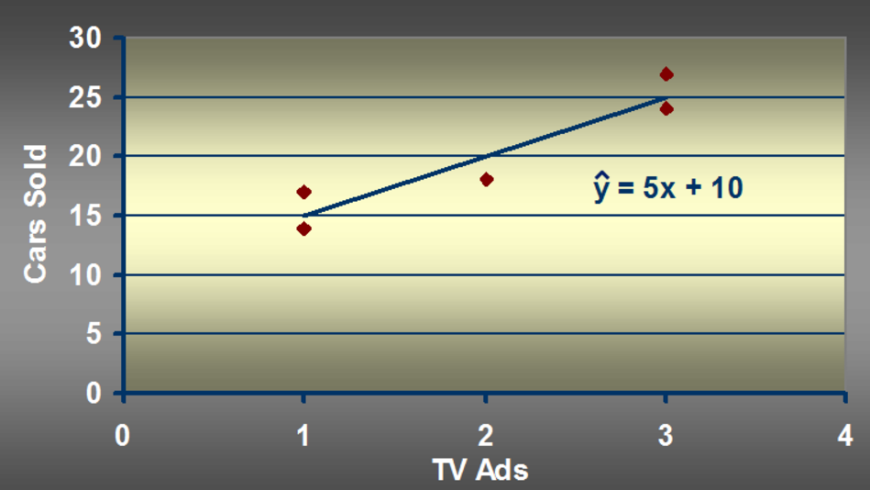
### （1）源数据

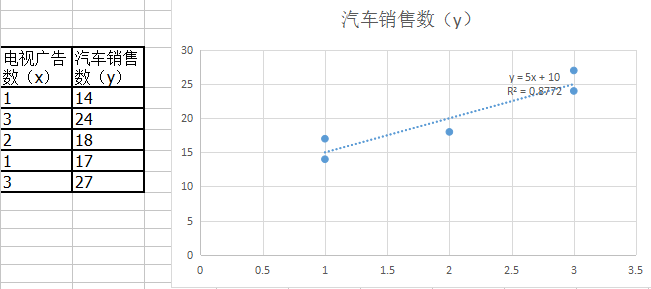
汽车卖家做电视广告数量与卖出的汽车数量：

|  |  |
| --- | --- |
| 电视广告数（x） | 汽车销售数（y） |
| 1 | 14 |
| 3 | 24 |
| 2 | 18 |
| 1 | 17 |
| 3 | 27 |
| Sum（x）=10，Avg（x）=2 | Sum（y）=100，Avg（y）=20 |

（2）找出能够拟合上述数据的最佳直线

1.1 如何找出适合简单线性回归模型的最佳回归线？





这里我们通过构造以下函数并使得该函数最小来求解最佳拟合的直线



如果使得上述函数最小，这里我们借助我们高中学习的导数方法，先求导并令导数为0来求解最佳的拟合直线的参数，如一次项的系数和常数项的值。

这里避免过于复杂的数学推导，直接给出一次项系数的参数和偏差项

，

1.2 计算

|  |  |
| --- | --- |
| 电视广告数（x） | 汽车销售数（y） |
| 1 | 14 |
| 3 | 24 |
| 2 | 18 |
| 1 | 17 |
| 3 | 27 |
| Sum（x）=10，Avg（x）=2 | Sum（y）=100，Avg（y）=20 |

分子 = (1-2)(14-20)+(3-2)(24-20)+(2-2)(18-20)+(1-2)(17-20)+(3-2)(27-20)

      = 6 + 4 + 0 + 3 + 7

      = 20

分母 = （1-2）^2 + (3-2)^2 + (2-2)^2 + (1-2)^2 + (3-2)^2

       = 1 + 1 + 0 + 1 + 1

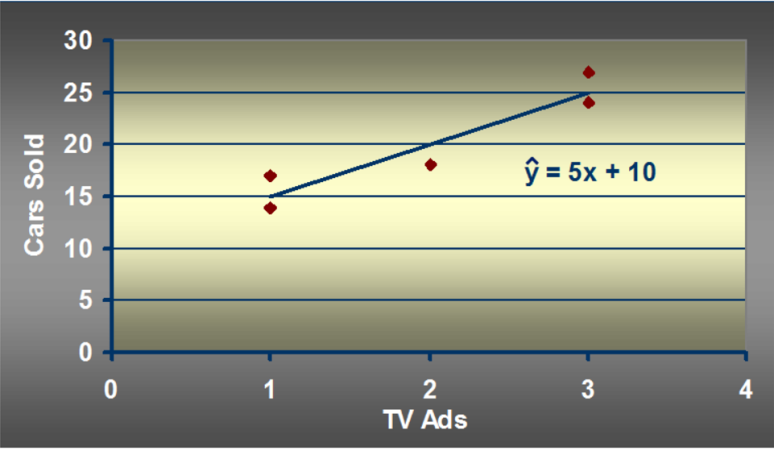
       4

b1 = 20/4  =5

b0=20-5\*2=10

### （2）预测

假设有一周广告数量为6，预测的汽车销售量是多少？



x\_given = 6

Y\_hat = 5\*6 + 10 = 40

### （3） Python实现

import numpy as np

def fitSLR(x, y):

    n = len(x)

    dinominator = 0

    numerator = 0

    for i in range(0, n):

        numerator += (x[i] - np.mean(x))\*(y[i] - np.mean(y))

        dinominator += (x[i] - np.mean(x))\*\*2

    b1 = numerator/float(dinominator)

    b0 = np.mean(y)/float(np.mean(x))

    return b0, b1

def predict(x, b0, b1):

    return b0 + x\*b1

x = [1, 3, 2, 1, 3]

y = [14, 24, 18, 17, 27]

b0, b1 = fitSLR(x, y)

print "intercept:", b0, " slope:", b1

x\_test = 6

y\_test = predict(6, b0, b1)

print "y\_test:", y\_test

# 多元线性回归

## 1.与简单线性回归区别(simple linear regression)

          多个自变量(x)

## 多元回归模型



## 3.多元回归方程



## 4.估计多元回归方程



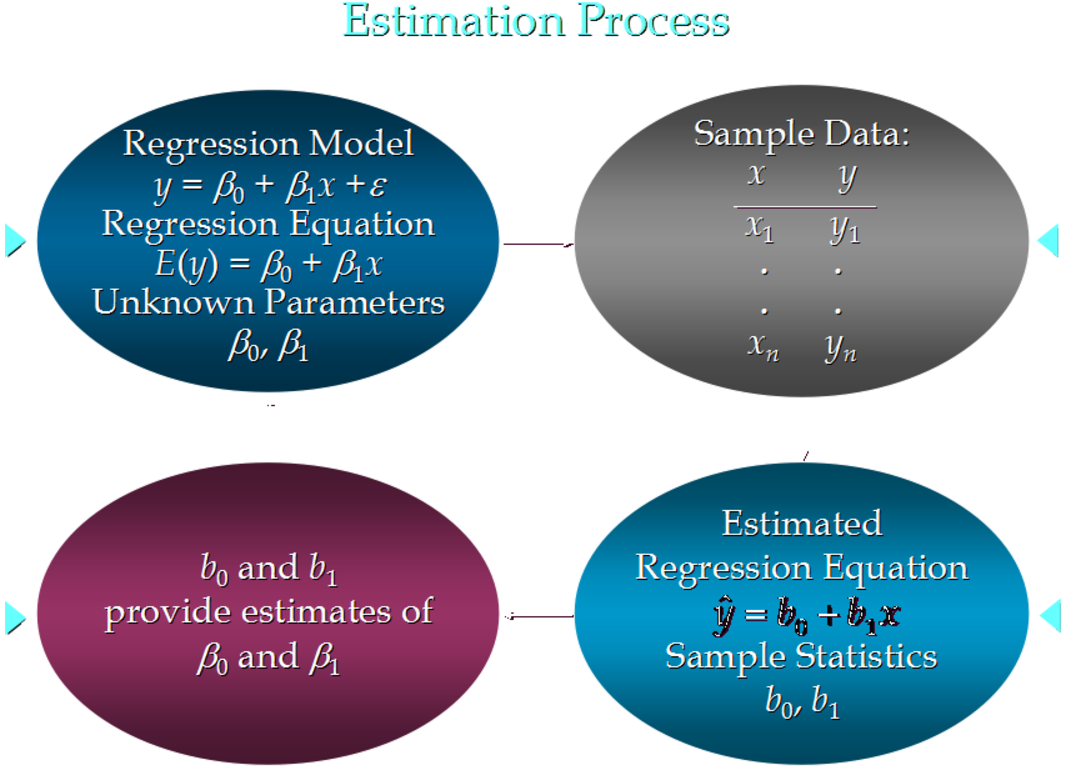
## 5. 估计方法---误差平方和最小

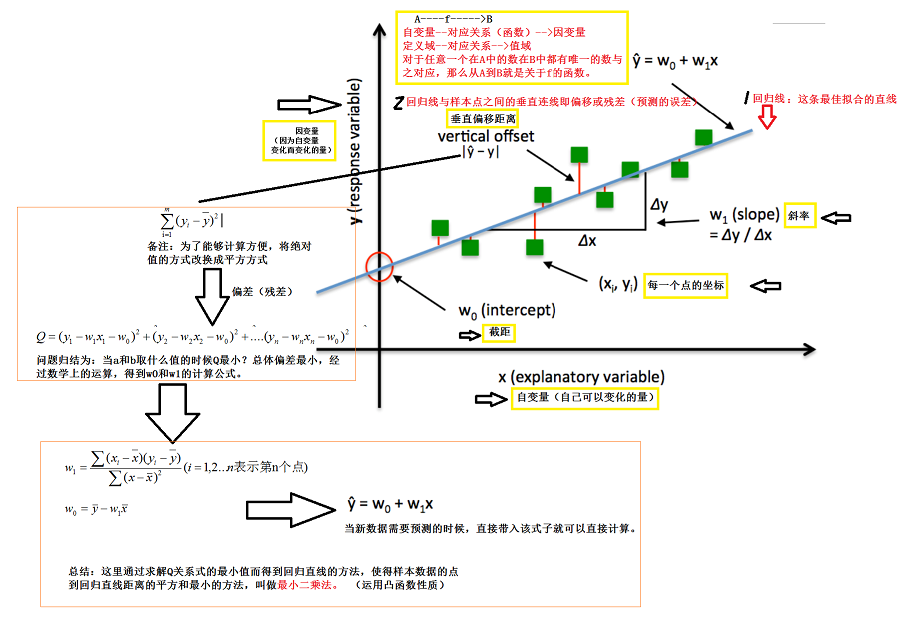
    使sum of squares（平方和）最小



    运算与简单线性回归类似，涉及到线性代数和矩阵代数的运算

## 6. 估计流程  (与简单线性回归类似）





## 例子

### 7.1描述

一家快递公司送货：X1： 运输里程 X2： 运输次数   Y：总运输时间

|  |  |  |  |
| --- | --- | --- | --- |
| **Driving**  **Assignment** | **X1=Miles**  **Traveled** | **X2=Number of Deliveries** | **Y= Travel Time (Hours)** |
| 1 | 100 | 4 | 9.3 |
| 2 | 50 | 3 | 4.8 |
| 3 | 100 | 4 | 8.9 |
| 4 | 100 | 2 | 6.5 |
| 5 | 50 | 2 | 4.2 |
| 6 | 80 | 2 | 6.2 |
| 7 | 75 | 3 | 7.4 |
| 8 | 65 | 4 | 6.0 |
| 9 | 90 | 3 | 7.6 |
| 10 | 90 | 2 | 6.1 |

Y(Time) = b0+ b1\*Miles + b2 \* Deliveries

Time = -0.869 + 0.0611 Miles + 0.923 Deliveries

### 7.2. 描述参数含义

     b1: 平均每多运送一英里，运输时间延长0.0611 小时

     b2: 平均每多一次运输，运输时间延长 0.923 小时

### 7.3. 预测

     如果一个运输任务是跑102英里，运输6次，预计多少小时？

     Time = -0.869 +0.0611 \*102+ 0.923 \* 6= 10.9 (小时）

### 7.4.处理分类型变量

如果自变量中有分类型变量(categorical data) , 如何处理？

|  |  |  |  |
| --- | --- | --- | --- |
| 英里数 | 次数 | 车型 | 时间 |
| 100 | 4 | 1 | 9.3 |
| 50 | 3 | 0 | 4.8 |
| 100 | 4 | 1 | 8.9 |
| 100 | 2 | 2 | 6.5 |
| 50 | 2 | 2 | 4.2 |
| 80 | 2 | 1 | 6.2 |
| 75 | 3 | 1 | 7.4 |
| 65 | 4 | 0 | 6 |
| 90 | 3 | 0 | 7.6 |

### 7.5.关于误差的分布

误差ε是一个随机变量，均值为0

ε的方差对于所有的自变量来说相等

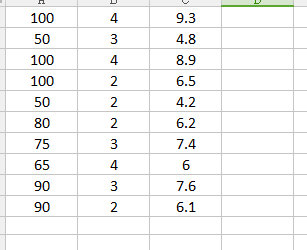
所有ε的值是独立的

ε满足正态分布，并且通过β0＋β１x1+β2x2+ ... +βpxp反映y的期望值

### 7.6.Python代码实战

数据：

1. 在execl中输入数据另存为csv文件
2. 直接在txt中输入数据以逗号隔开



Python代码：

#-\*-coding:utf8-\*-  
import numpy as np  
from numpy import genfromtxt  
from sklearn import datasets,linear\_model  
#组织数据，通过逗号分隔的CSV文件组织  
datapath=r"D:\delivery.csv"  
delivesyData=genfromtxt(datapath,delimiter=',')  
print "data",delivesyData  
#区分特征和类标签  
X=delivesyData[:,:-1]  
Y=delivesyData[:,-1]  
#打印切分的数据  
print "X",X  
print "Y",Y  
#模型初始化  
regr=linear\_model.LinearRegression()  
regr.fit(X,Y)  
#打印模型的属性  
print "coefficient",regr.coef\_  
print "intercept",regr.intercept\_  
#适用模型进行预测  
xPred=[102,6]  
yPred=regr.predict(xPred)  
print "Y-predicted",yPred

## 8性能评估详解：

不同于类别预测，不能苛刻回归预测的数值结果要严格的和真实值一致。一般情况下，我们希望衡量预测值和真实值之间的差距。因此，可以通过多种测评函数进行评价。

通常最为直观的评价指标包括：

表示回归模型的预测结果，表示样本的真实结果值。

1. 平均绝对误差：





1. 均方误差：





1. R-Squared：

既考量了回归值与真实值的差异，同时也兼顾了问题的真实值的变动，f(x)代表了回归模型x的预测值。





代表测试数据真实值的方差（内部差异）；代表了回归值与真实值之间的平差差异(回归差异)。因此在统计含以上，R-Squared用来衡量模型回归结果的波动可被真实值验证的百分比，也暗示了模型在数值回归方面的能力。

# 极大似然估计和高斯分布推导最小二乘

似然函数：

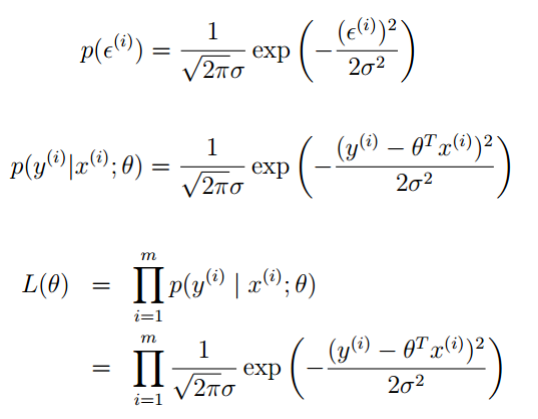
第i个样本的x和第i个样本的预测值y，是误差，表示的是第i个样本的真实值和预测值之间的差距，比如房子的预测价格和实际价格。

在实际应用中都是独立同分布的，**服从**均值是0，方差是sima\*\*2的高斯分布（正态分布）。利用房屋售价理解，所有房屋价格的平均值就是目前房屋的正常的价格范围。

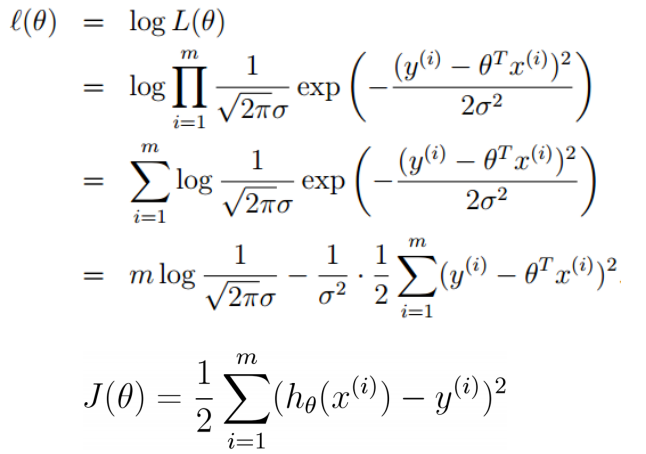
服从正态分布的原因是根据中心极限定理，在实际问题中很多现象都可以看作众多因素独立影响的综合反映，往往近似服从正态分布。如城市耗电量，就是大量用户耗电量总和，每家每户的耗电量可以看作是所有用户的平均值。

模型：

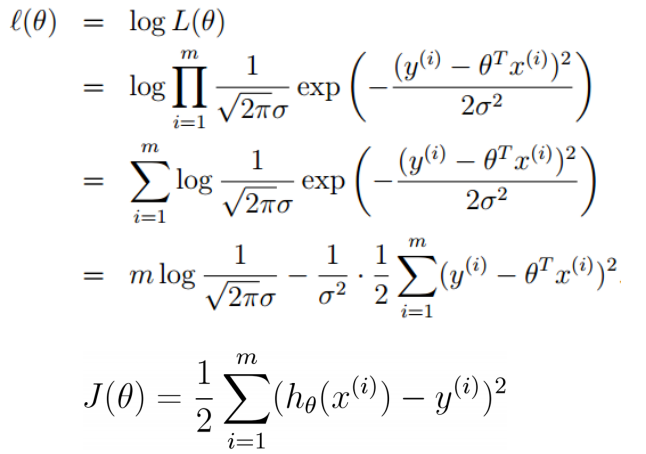
可以用正态分布表示，从而得到y的概率密度函数，使用最大似然估计（假设各变量之间是独立的）：



对数似然函数的建立：

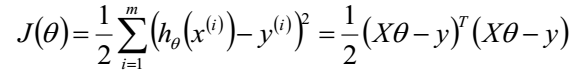


目标（损失）函数变为：求J的最小值

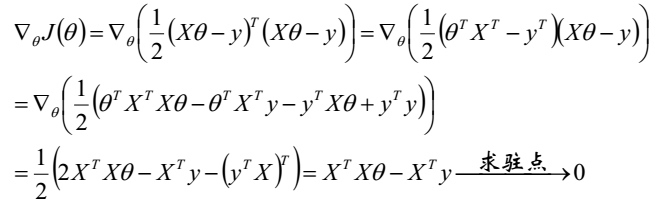


求J的最小值，也就是求解参数theta的最优解，求解出来之后带入到模型中就得到了最终想要的模型。

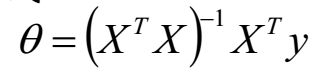
目标函数形式：



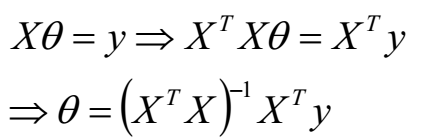
求梯度：



参数theta：

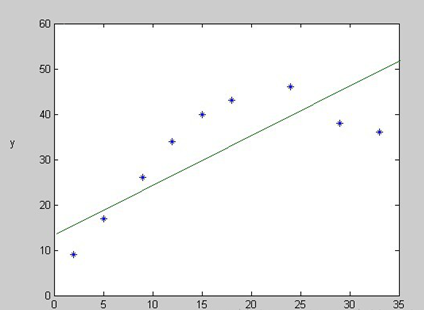


方便记忆的方式：（仅用于计算，一般x\*theta+lambda=y，lambda是误差，实际值和误差值不可能完全相等）



# 局部加权回归

当目标假设是线性模型时，使用线性回归自然能够很好拟合数据，但如果目标假设不是线性模型，比如一个忽上忽下的函数，这时用线性模型就拟合的很差。比如下面坐标表示的情形，用一条直线来拟合图上的点显然时不合适的。



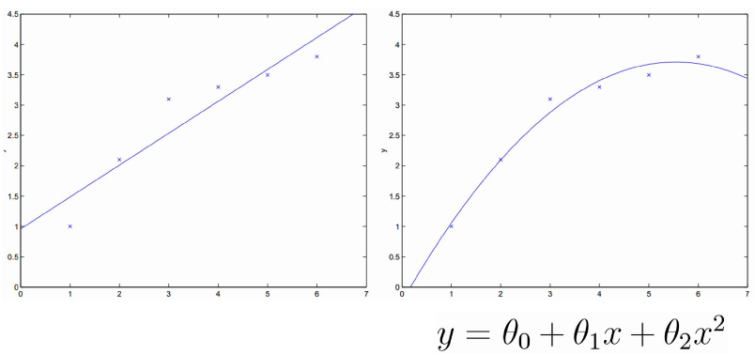
这种情况叫欠拟合（underfitting）

方法1：通过增大模型的复杂度，可以用一个二次函数做拟合

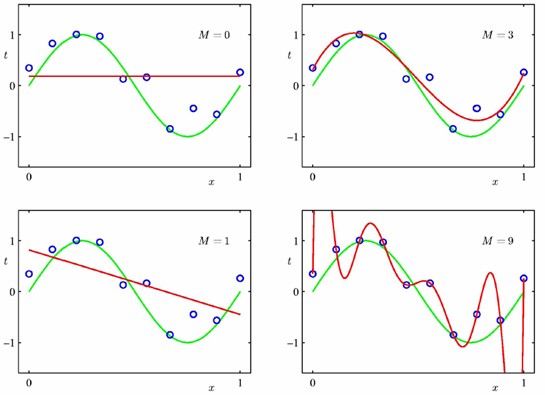
方法2：局部加权线性回归

**方法1理解：**

通常来说，对待这种情况，特征的选择可能会非常重要，比如，对于下面的图，如果加特征：x的平方或者比如加sin(x)可能能够实现很好的拟合这些点。多项式拟合能拟合所有数据（泰勒公式），但是在预测新样本的时候表现的又会变得很差很糟糕，因为它导致数据的过拟合（overfitting），不符合数据的真实的模型。



## 回顾过拟合与欠拟合

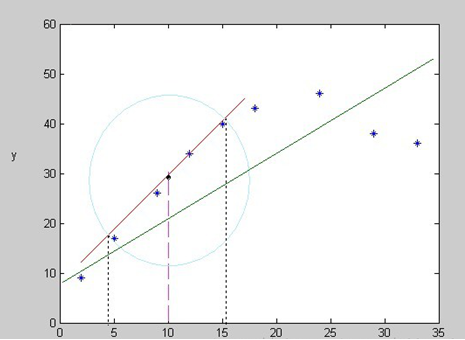


上图给出了M=0，M=1，M=3，M=9时的多项式函数拟合的情况，其中绿色曲线为真模型，红色为预测模型。

## 2.采用局部加权解决模型过拟合

针对过拟合的情况，引入局部加权方式解决。

局部加权线性回归(locally weighted linear regression)也就是在“局部”采用线性回归。



对上图的说明：假如给定x=10，要预测对应的y值，如果用线性拟合的方法，得到的是图中的绿线，有很多点没有在绿线上，误差较大。

但只考虑两条虚线之间，也就是圆圈之内的四个点进行线性拟合，那么得到的是那条红线，对应的点在这条红线上相对的y值就是预测值，从直观上来看就可以认为预测会更准确。

在待测点附近小子集上以最小均方误差来进行普通线性回归，我们使用 实现上述加权方案。

如何理解？为了实现小子集的线性回归，只取这一小部分进行线性回归。怎么只取这一小部分数据呢？这里就是通过控制，在待测点附近取1，较远的地方取0，那么就能够实现带测点附近小子集的线性回归。

当然上述我们并不严格取1或0，我们可以用近似的1和0来近似来代替。是权值，它的作用是在于**根据要预测的点与数据集中的点距离来为数据集中的点赋予权值**，当某点距离待预测点较大的时候，其权重较小，否则较大。

注意：w权值通常去的是距离分之1.

## 非参数学习方法

局部加权线性回归（Locally Weight Linear Regression，WLR）是一种非参数学习方法，为什么局部加权回归又叫非参数学习方法呢？

参数学习方法是这样一种方法：在训练完成数据之后得到一系列的训练参数，然后根据训练参数来预测新样本的值，这是不在依赖之前的训练数据了，参数值是确定的。而非参数学习方法是这样的一种算法：在预测新样本的时候每次都会重新训练数据得到最新的参数值，也就是每次预测新样本都会依赖训练数据集合，所以每次得到的参数是不确定的。

# 正则化

## 5.1正则化

模型选择的典型方法就是正则化，正则化的一般形式如下：



其中，第一项是经验风险，也就是常见的平方差损失函数，第二项是正则项，正则化项可以取不同的形式，例如，正则化项可以使模型参数的范数。在回归问题中，损失函数是平方损失，正则化项可以使参数向量的L2范数或者L1范数：





使用正则化罚项可以对不重要的属性或影响模型准确度的属性进行惩罚。

## 5.2L1和L2正则的区别，如何选择L1和L2正则？

### 5.2.1三者的概念

L0正则化的值是**模型参数中非零参数的个数**。



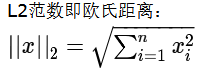
也就是如果我们使用L0范数，即希望w的大部分元素都是0。 （w是稀疏的）所以可以用于ML中做稀疏编码，特征选择。通过最小化L0范数，来寻找最少最优的稀疏特征项。但不幸的是，L0范数的最优化问题是一个NP hard问题，而且理论上有证明，L1范数是L0范数的最优凸近似，因此通常使用L1范数来代替。因此很少用L0正则化，但是在面试中会常用到。

L1正则化表示各个参数绝对值之和。



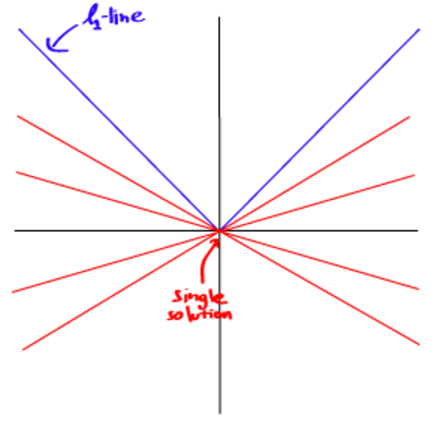
L1范数的解通常是稀疏性的，倾向于选择数目较少的一些非常大的值或者数目较多的insignificant的小值。

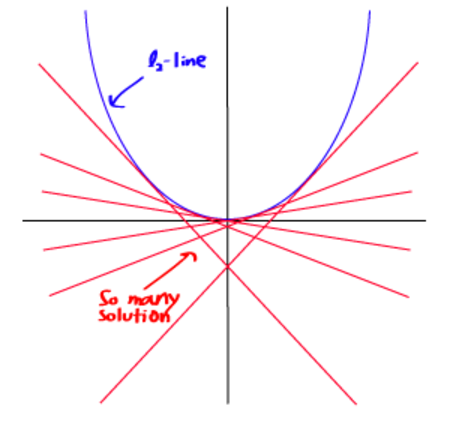
L2正则化标识各个参数的平方的和的开方值。



L2范数越小，可以使得w的每个元素都很小，接近于0，但L1范数不同的是他不会让它等于0而是接近于0.

### 5.2.2L1范数与L2范数的比较





但由于L1范数并没有平滑的函数表示，起初L1最优化问题解决起来非常困难，但随着计算机技术的到来，利用很多凸优化算法使得L1最优化成为可能。

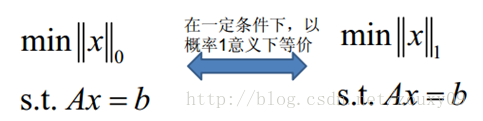
### 5.2.3三者比较

L1 Norm 和L2 Norm的区别（核心：L2对大数，对outlier更敏感！）：

L1优点是能够获得sparse（稀疏）模型，对于large-scale的问题来说这一点很重要，因为可以减少存储空间。缺点是加入L1后目标函数在原点不可导，需要做特殊处理。  
 L2优点是实现简单，能够起到正则化的作用。缺点就是L1的优点：无法获得sparse模型。实际上L1也是一种妥协的做法，要获得真正sparse的模型，要用L0正则化。  
 L0范数是指向量中非0的元素的个数。如果我们用L0范数来规则化一个参数矩阵W的话，就是希望W的大部分元素都是0（参数W是稀疏的）。

   L1范数是指向量中各个元素绝对值之和。

既然L0可以实现稀疏，为什么不用L0，而要用L1呢？**个人理解一是因为L0范数很难优化求解（NP难问题），二是L1范数是L0范数的最优凸近似，而且它比L0范数要容易优化求解。**所以大家才把目光和万千宠爱转于L1范数。



**（总结）**来个一句话**总结**：L1范数和L0范数可以实现稀疏，L1因具有比L0更好的优化求解特性而被广泛应用。

除了L1范数，还有一种更受宠幸的规则化范数是L2范数: ||W||2。它也不逊于L1范数，它有两个美称，在回归里面，有人把有它的回归叫“岭回归”（Ridge Regression），有人也叫它“权值衰减weight decay”。这用的很多吧，因为它的强大功效是改善机器学习里面一个非常重要的问题：过拟合。

# 岭回归

**岭回归是基于L2罚项的模型，只是在最小二乘代价函数中加入权重的平方和。**

线性回归的目标函数为：



岭回归的目标函数：



通过增加超参数的值，我们可以增加正则化的强度，同时，也就降低了权重对模型的影响。一定要注意：正则化项不影响截距

**更改为补充的：**

如果数据的特征比样本点多的场景我们怎么办？是否还可以用线性回归和局部加权回归方法呢？答案是否定的，因为线性回归和局部加权回归是需要存在的前提下。而当数据的特征数比样本点多的时候我们无法保证存在，因此，为了达到进行回归预测的目的，我们引入岭回归，即在原目标函数的基础上加入一个，使得矩阵变成非奇异矩阵，进而能对求逆。其中I是一个m\*m的单位矩阵，对角上元素全为1，其他元素全为0，而是自定义的数值，在这种情况下回归系数的计算公式将变成：



### 1.岭回归的目标函数

接下来探索下岭回归的目标函数。

大家还记得，线性回归的目标函数为：



我们优化线性回归的目标函数为：



为了防止过拟合，增加正则化项，目标函数变为



上述目标函数等价于





对其进行求导，得到：



令导数为0，得



上面得到的就是岭回归公式。

为了实现岭回归和缩减技术，首先需要对特征进行标准化处理，在使用上述公式进行岭回归求解。

常用得标准化方法，公式如下，样本值减去均值除以方差值



### 3.总结

岭回归最先是用来处理**特征数大于样本数**得情况，现在也用于在估计中加入偏差，从而得到更好的估计，即控制模型参数，结构风险最小化。这里通过引入来限制了所有w之和，通过引入该惩罚项，能够减少不重要得参数，这个技术在统计学中也叫做缩减。

岭回归使用了单位矩阵乘以常量，我们观察其中得单位矩阵I，我们看到值1贯穿整个对角线，其余元素全是0.形象地，在0构成得平面上有一条1组成得“岭”，这就是岭回归中得“岭”得由来。

### 4.岭回归优点：

1. 在特征数M大于样本数N时，不可逆，故不能直接用LR，而岭回归就可以。
2. 通过引入惩罚项，防止过拟合。

# Lasso回归和弹性网络(ElasticNet)

## 7.1Lasso回归

对于稀疏数据训练的模型，还有另一种解决方案，即LASSO，基于正则化项的强度，某些权重可以变为0，这也使得Lasso称为监督学习的一种特征选择技术。

Lasso的正则化项时L1正则，目标函数为



**注：**

Lasso的正则化项时L1正则，目标函数为



等价于



这个细微的变化，加大增加了计算复杂度，因为其不可直接求导。

Lasso回归系数：

不同于岭回归系数：

因为使用普通的最小二乘法回归在两个或更多的特征相关时，可能会得出一个很大的正系数和一个很大的负系数。可以参考岭回归图中当参数足够小的时候，岭回归变化不大，但是使用Lasso回归一些系数会因此被迫缩减到0，这个特征能够使得我们很好的理解数据。在Lasso回归中可以避免系数过大过小的问题。

针对不可求导的问题，如果要在新的约束条件下解出回归系数，需要使用二次规划算法，为此我们讲解一种更为简便的算法解决该问题，该方法为前向逐步回归。

## 7.2弹性网络

弹性网络是介于岭回归和LASSO之间的一个折中，其中包含了一个用于稀疏化的L1罚项，以及一个消除Lasso限制（比如筛选变量的数量）的L2罚项。



## 7.3sklearn的实现

岭回归的初始化方式

岭回归：

from sklearn.linear\_model import Ridge

ridge=Ridge(alpha=1.0)

#正则化程度是通过alpha来调节的，类似于参数x

LASSO回归：

from sklearn.linear\_model import Lasso

lasso=Lasso(alpha=1.0)

#正则化程度是通过alpha来调节的，类似于参数x

弹性网络ElasticNet：

from sklearn.linear\_model import ElasticNet

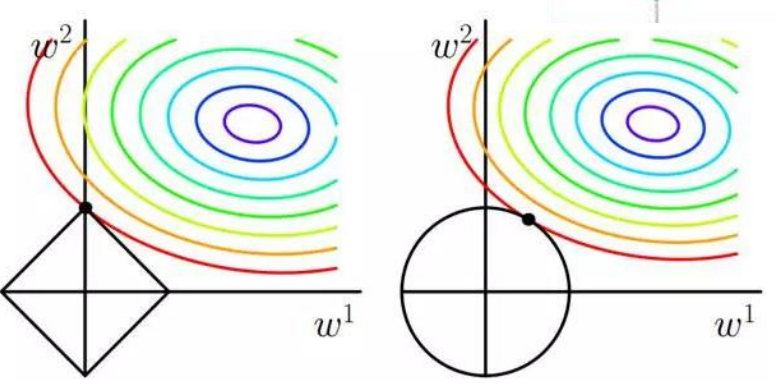
lasso=ElasticNet(alpha=1.0,l1\_ratio=0.5)

#l1\_ratio是L1正则化参数，alpha是L2的正则化参数

#当l1\_ratio=1.0此时就转化为了lasso

## 7.4总结

LASSO相比Ridge回归可以进行特征选择，如下图。

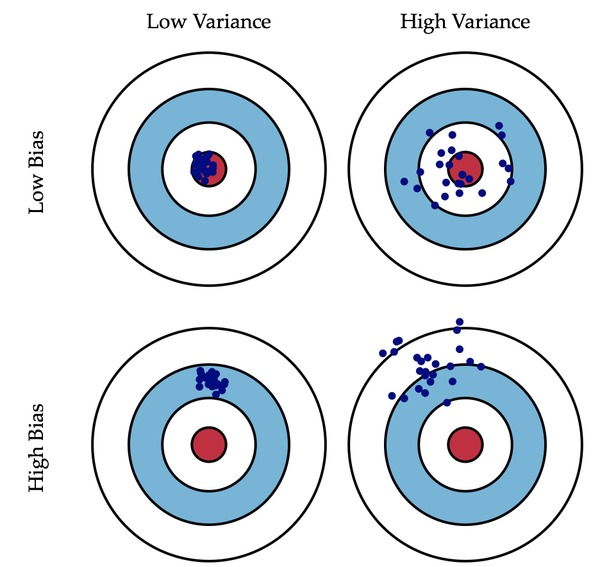


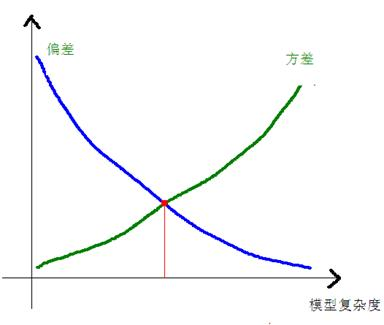
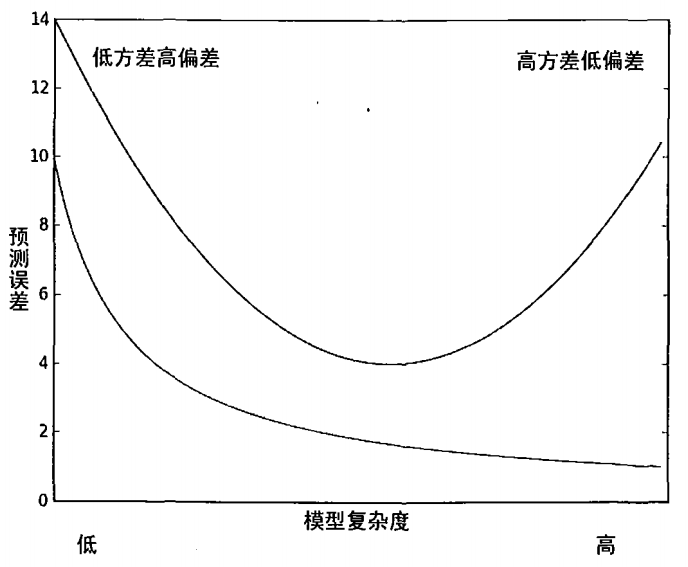
使用岭回归和Lasso回归，都对系数做了限制，模型也就增加了模型的偏差bias，与此同时却减少了模型的方差。下面我们权衡数据的偏差和方差。

# 权衡偏差与方差

偏差：描述的是预测值（估计值）的期望与真实值之间的差距。偏差越大，越偏离真实数据，如下图第二行所示。

方差：描述的是预测值的变化范围，离散程度，也就是离其期望值的距离。方差越大，数据的分布越分散，如下图右列所示。





利用前面的岭回归和Lasso回归可以将一些系数缩减为0，这时是增加了模型偏差，但是减少了模型的复杂，试想有些特征的系数被缩减为0，该特征就不会在线性拟合的时候起作用，就达到了属性约简，减少模型复杂度的目的。图中左侧是参数缩减过于严厉的结果，而右侧是无缩减的效果。属性无缩减的时候，模型的复杂度就比较搞了。

偏差方差折中与测试误差及训练误差的关系。右侧的曲线就是测试误差，在中间部分最低。为了做出最好的预测，我们应该调整模型复杂度（属性约简）来达到测试误差的最小值。

# 波斯顿房价问题的回归预测（纵向）

## 思路

假设有一个房屋销售的数据如下：

面积(m2) 销售价钱（万元）

123 250

150 320

87 160

102 220

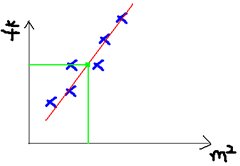
… …

我们可以画一个图，x轴是房屋的面积。y轴是房屋的售价：

[](http://images.cnblogs.com/cnblogs_com/LeftNotEasy/WindowsLiveWriter/1_1270E/image_2.png)

如果有一个新的面积，假设在销售价钱的记录中没有的，而我们又想知道房屋的售卖价格，我们怎么办呢？图中绿色的点就是我们想要预测的点。

假设我们知道了红色的这条直线，那么给出房屋的面积，马上就可以给出房屋的售价。因此，我们需要找到这样的一条红色直线。

[](http://images.cnblogs.com/cnblogs_com/LeftNotEasy/WindowsLiveWriter/1_1270E/image_6.png)

回顾：

在实际案例学习之前，一定要**搞清楚回归问题和分类问题的**差别：待预测的目标是否为连续变量。

回顾模型：

相信大家在分类模型中已经注意到了为了将原本在实数域上的计算结果映射到(0,1)之间，引入了logistic函数（也就是给大家讲的sigmod函数）。而在线性回归中，由于预测目标直接是实数域上的数值，因此优化的目标就是最小化预测结果与真实值之间的差异。

数学表达：

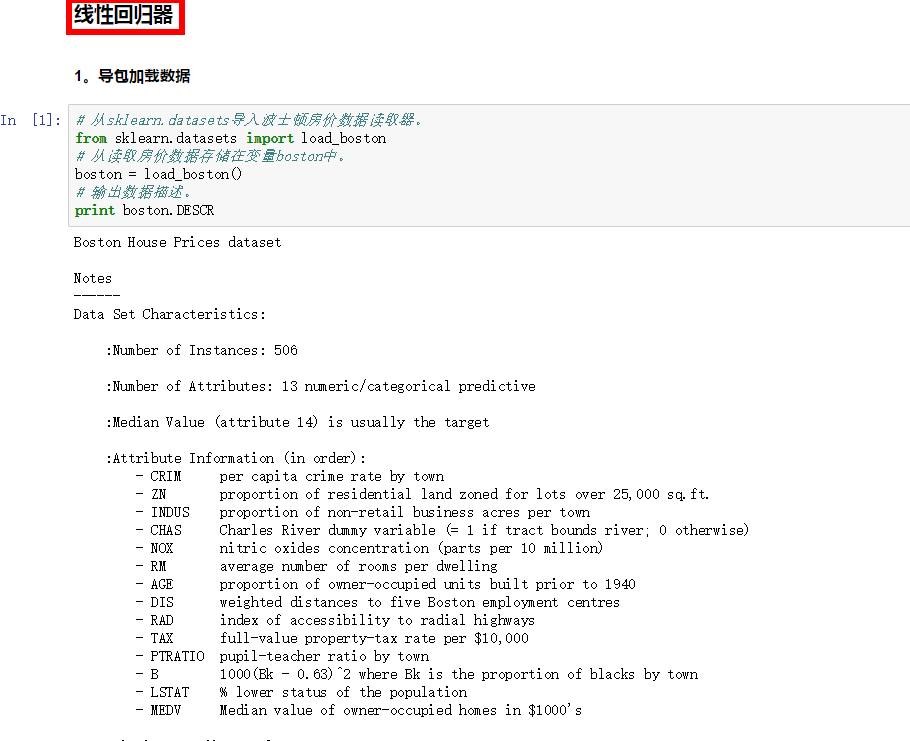
* 有一组m个用于训练的特征向量：
* 以及对应的回归目标：
* 使用最小二乘预测损失L(w,b)
* 线性回归模型优化目标的式子：
  + 
* 为了学习到决定模型的参数，即w和b，一般有两种方法（1）精确计算解析算法和（2）快速的随机梯度下降法（SGD）估计算法。

**注意：不管是随机梯度上升（SGA）还是梯度下降（SGD），都属于用梯度法迭代估计参数的过程。梯度上升用于目标最大化，梯度下降用于目标最小化。**

该部分请着重比较各种回归模型的性能与优缺点的比较。

## 导包和加载数据

500行记录，每行数据有13项数值型特征描述和一个类标签（目标房价），另外该数据没有缺失的属性或特征值，分析起来更加方便。

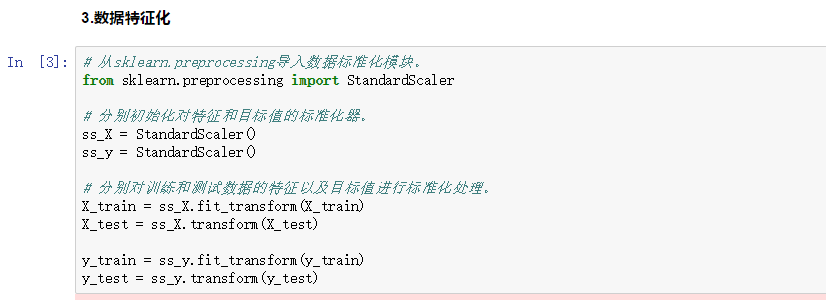


## 数据切分

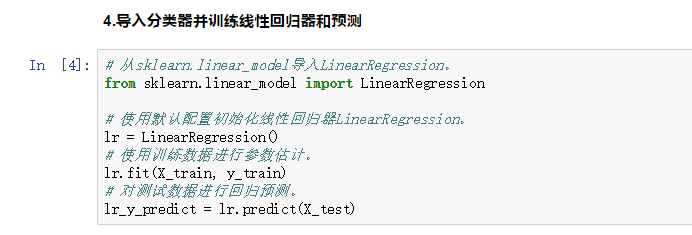
通过下面对数据的初步分析之后，预测目标房价之间的差异较大，从5-50不等，因此需要对特征以及目标值进行标准化处理。



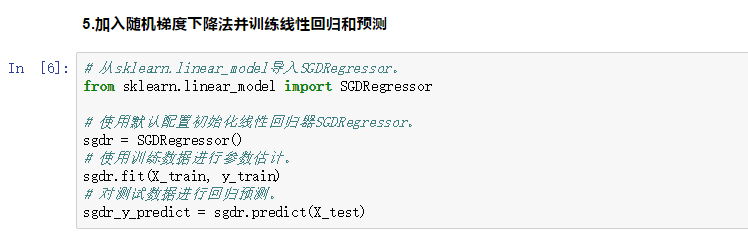
## 特征工程



## 线性回归器预测房价



## 5.随机梯度下降法的回归器预测房价



## 线性模型的评估

性能评估：

不同于类别预测，不能苛刻回归预测的数值结果要严格的和真实值一致。一般情况下，我们希望衡量预测值和真实值之间的差距。因此，可以通过多种测评函数进行评价。

通常最为直观的评价指标包括：

表示回归模型的预测结果，表示样本的真实结果值。

1. 平均绝对误差：





1. 均方误差





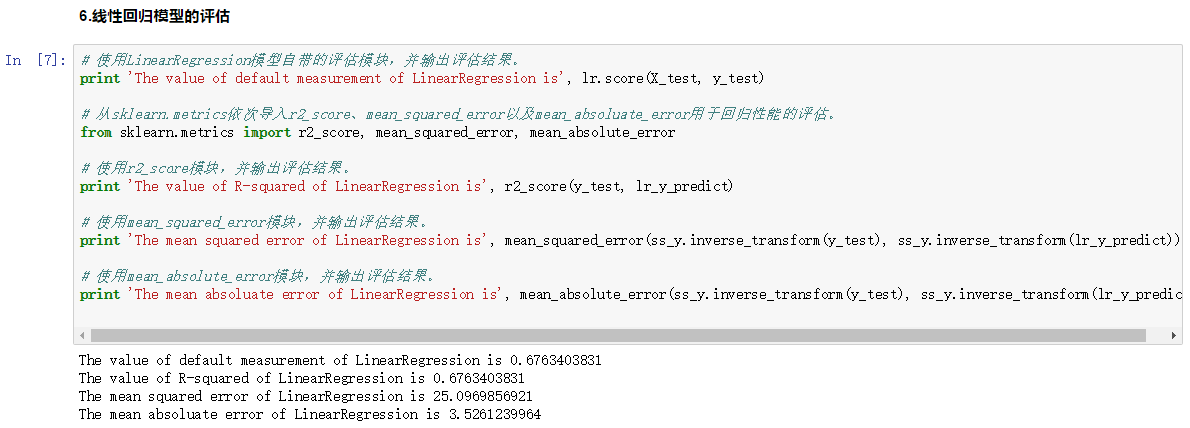
1. R-Squared：

既考量了回归值与真实值的差异，同时也兼顾了问题的真实值的变动，f(x)代表了回归模型x的预测值。

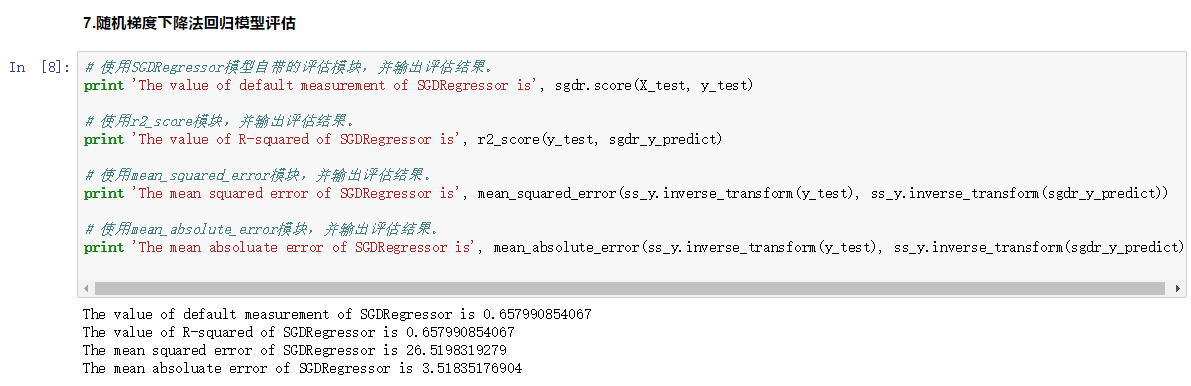




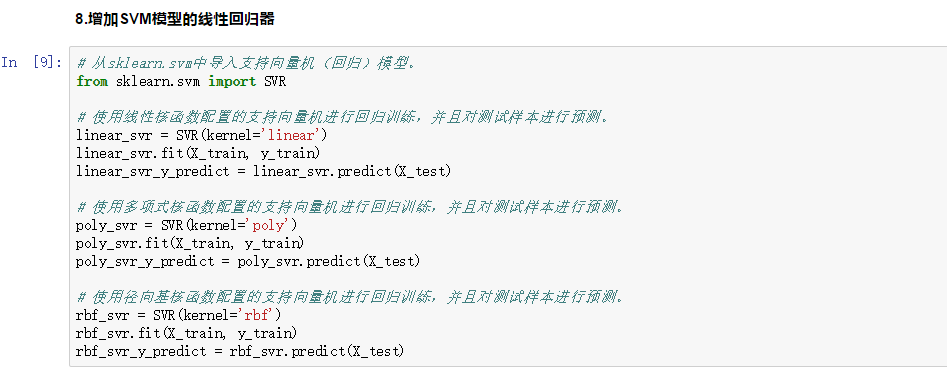
代表测试数据真实值的方差（内部差异）；代表了回归值与真实值之间的平差差异(回归差异)。因此在统计含以上，R-Squared用来衡量模型回归结果的波动可被真实值验证的百分比，也暗示了模型在数值回归方面的能力。



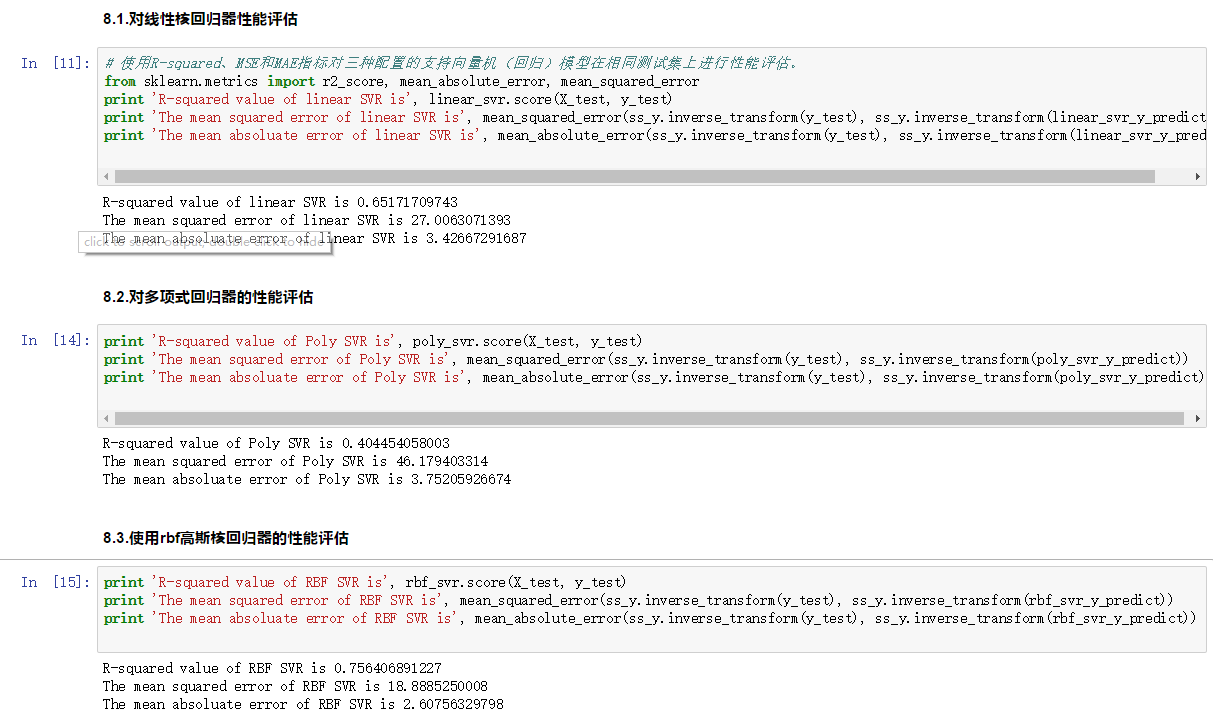
## 随机梯度下降法的评估



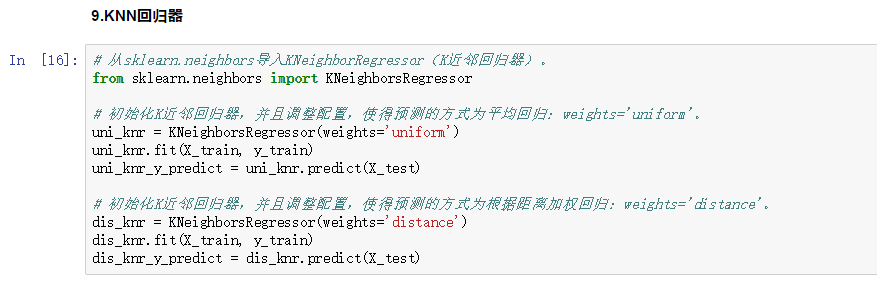
## SVM回归器



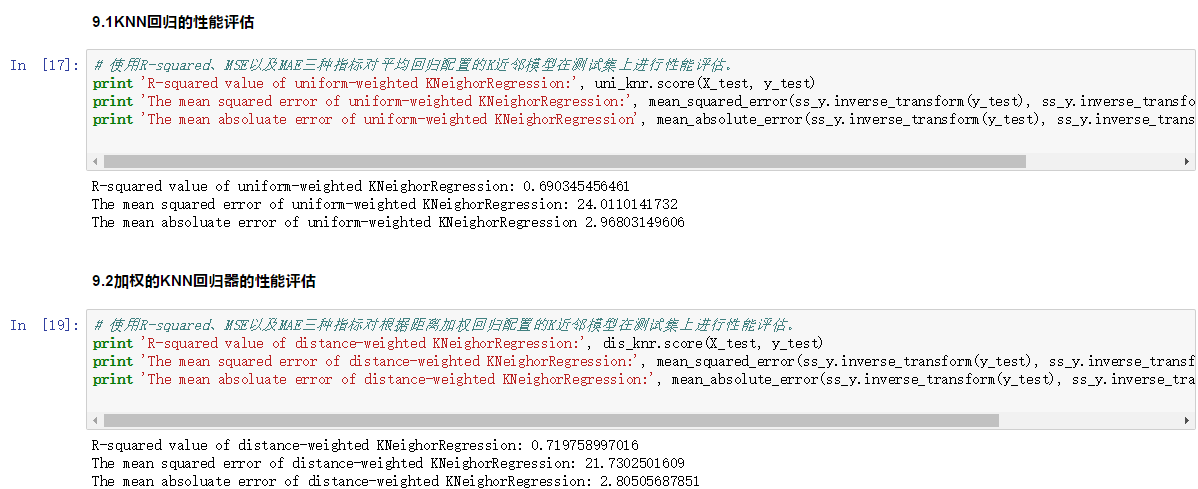
## SVM回归器评估



## KNN回归器



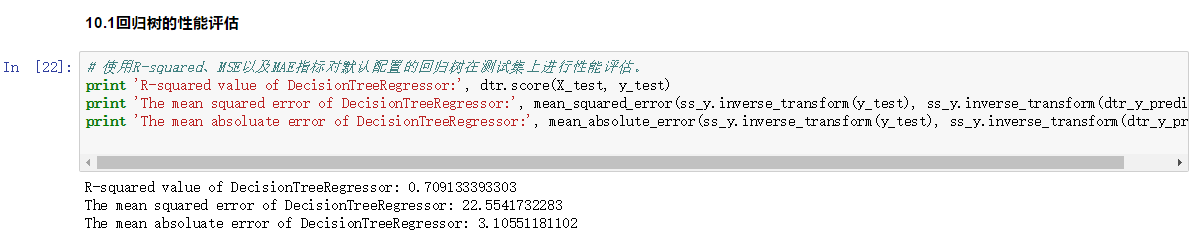
## KNN回归器的评估



## 回归树



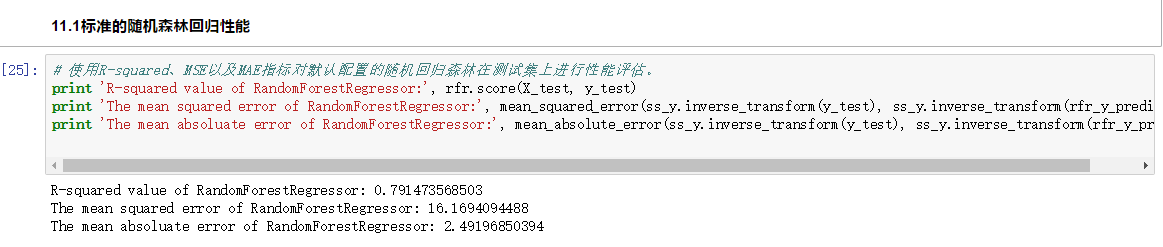
## 回归树性能评估

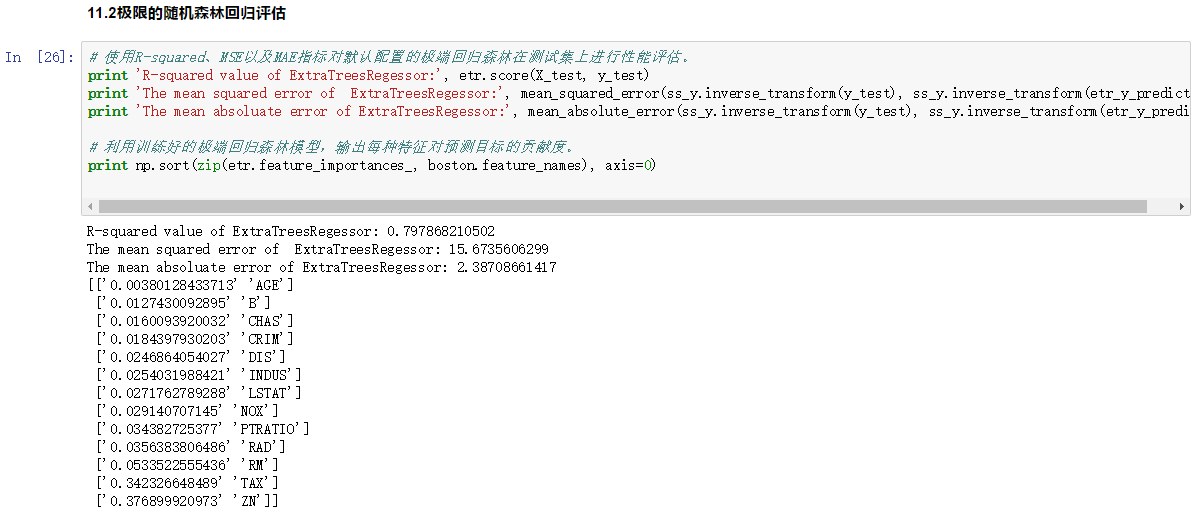


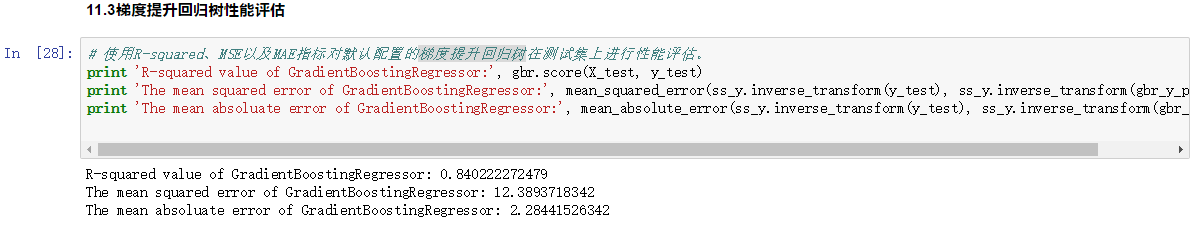
## 14随机森林回归树



## 15随机森林回归的评估







# 十、线性回归的sklearn实战（练习题）

## 线性回归sklearn案例

# -\*- coding: utf-8 -\*-

"""

广义线性模型

~~~~~~~~~~~~~~~~~~~~~~~~~~

LinearRegression

:copyright: (c) 2016 by the huaxz1986.

:license: lgpl-3.0, see LICENSE for more details.

"""

import matplotlib.pyplot as plt

import numpy as np

from sklearn import datasets, linear\_model,cross\_validation

def load\_data():

'''

加载用于回归问题的数据集

:return: 一个元组，用于回归问题。元组元素依次为：训练样本集、测试样本集、训练样本集对应的值、测试样本集对应的值

'''

diabetes = datasets.load\_diabetes()#使用 scikit-learn 自带的一个糖尿病病人的数据集

return cross\_validation.train\_test\_split(datasets.data,diabetes.target,

test\_size=0.25,random\_state=0) # 拆分成训练集和测试集，测试集大小为原始数据集大小的 1/4

def test\_LinearRegression(\*data):

'''

测试 LinearRegression 的用法

:param data: 可变参数。它是一个元组，这里要求其元素依次为：训练样本集、测试样本集、训练样本的值、测试样本的值

:return: None

'''

X\_train,X\_test,y\_train,y\_test=data

regr = linear\_model.LinearRegression()

regr.fit(X\_train, y\_train)

print('Coefficients:%s, intercept %.2f'%(regr.coef\_,regr.intercept\_))

print("Residual sum of squares: %.2f"% np.mean((regr.predict(X\_test) - y\_test) \*\* 2))

print('Score: %.2f' % regr.score(X\_test, y\_test))

if \_\_name\_\_=='\_\_main\_\_':

X\_train,X\_test,y\_train,y\_test=load\_data() # 产生用于回归问题的数据集

test\_LinearRegression(X\_train,X\_test,y\_train,y\_test) # 调用 test\_LinearRegression

## 岭回归

# -\*- coding: utf-8 -\*-  
"""  
 广义线性模型:岭回归  
"""  
import matplotlib.pyplot as plt  
import numpy as np  
from sklearn import datasets, linear\_model,cross\_validation  
  
def load\_data():  
 '''  
 加载用于回归问题的数据集  
 :return: 一个元组，用于回归问题。元组元素依次为：训练样本集、测试样本集、训练样本集对应的值、测试样本集对应的值  
 '''  
 diabetes = datasets.load\_diabetes()#使用 scikit-learn 自带的一个糖尿病病人的数据集  
 return cross\_validation.train\_test\_split(diabetes.data,diabetes.target,  
 test\_size=0.25,random\_state=0) # 拆分成训练集和测试集，测试集大小为原始数据集大小的 1/4  
  
def test\_Ridge(\*data):  
 '''  
 测试 Ridge 的用法  
 :param data: 可变参数。它是一个元组，这里要求其元素依次为：训练样本集、测试样本集、训练样本的值、测试样本的值  
 :return: None  
 '''  
 X\_train,X\_test,y\_train,y\_test=data  
 regr = linear\_model.Ridge()  
 regr.fit(X\_train, y\_train)  
 print('Coefficients:%s, intercept %.2f'%(regr.coef\_,regr.intercept\_))  
 print("Residual sum of squares: %.2f"% np.mean((regr.predict(X\_test) - y\_test) \*\* 2))  
 print('Score: %.2f' % regr.score(X\_test, y\_test))  
def test\_Ridge\_alpha(\*data):  
 '''  
 测试 Ridge 的预测性能随 alpha 参数的影响  
 :param data: 可变参数。它是一个元组，这里要求其元素依次为：训练样本集、测试样本集、训练样本的值、测试样本的值  
 :return: None  
 '''  
 X\_train,X\_test,y\_train,y\_test=data  
 alphas=[0.01,0.02,0.05,0.1,0.2,0.5,1,2,5,10,20,50,100,200,500,1000]  
 scores=[]  
 for i,alpha in enumerate(alphas):  
 regr = linear\_model.Ridge(alpha=alpha)  
 regr.fit(X\_train, y\_train)  
 scores.append(regr.score(X\_test, y\_test))  
 ## 绘图  
 fig=plt.figure()  
 ax=fig.add\_subplot(1,1,1)  
 ax.plot(alphas,scores)  
 ax.set\_xlabel(r"$\alpha$")  
 ax.set\_ylabel(r"score")  
 ax.set\_xscale('log')  
 ax.set\_title("Ridge")  
 plt.show()  
if \_\_name\_\_=='\_\_main\_\_':  
 X\_train,X\_test,y\_train,y\_test=load\_data() # 产生用于回归问题的数据集  
 test\_Ridge(X\_train,X\_test,y\_train,y\_test) # 调用 test\_Ridge  
 #下面检验不同的a值对于预测性能的影响，通过绘图函数可以看到结果  
 # test\_Ridge\_alpha(X\_train,X\_test,y\_train,y\_test) # 调用 test\_Ridge\_alpha

## lasso回归

# -\*- coding: utf-8 -\*-  
"""  
 广义线性模：Lasso  
  
"""  
import matplotlib.pyplot as plt  
import numpy as np  
from sklearn import datasets, linear\_model,cross\_validation  
def load\_data():  
 '''  
 加载用于回归问题的数据集  
 :return: 一个元组，用于回归问题。元组元素依次为：训练样本集、测试样本集、训练样本集对应的值、测试样本集对应的值  
 '''  
 diabetes = datasets.load\_diabetes()#使用 scikit-learn 自带的一个糖尿病病人的数据集  
 return cross\_validation.train\_test\_split(diabetes.data,diabetes.target,  
 test\_size=0.25,random\_state=0) # 拆分成训练集和测试集，测试集大小为原始数据集大小的 1/4  
def test\_Lasso(\*data):  
 '''  
 测试 Lasso 的用法  
 :param data: 可变参数。它是一个元组，这里要求其元素依次为：训练样本集、测试样本集、训练样本的值、测试样本的值  
 :return: None  
 '''  
 X\_train,X\_test,y\_train,y\_test=data  
 regr = linear\_model.Lasso()  
 regr.fit(X\_train, y\_train)  
 print('Coefficients:%s, intercept %.2f'%(regr.coef\_,regr.intercept\_))  
 print("Residual sum of squares: %.2f"% np.mean((regr.predict(X\_test) - y\_test) \*\* 2))  
 print('Score: %.2f' % regr.score(X\_test, y\_test))  
def test\_Lasso\_alpha(\*data):  
 '''  
 测试 Lasso 的预测性能随 alpha 参数的影响  
 :param data: 可变参数。它是一个元组，这里要求其元素依次为：训练样本集、测试样本集、训练样本的值、测试样本的值  
 :return: None  
 '''  
 X\_train,X\_test,y\_train,y\_test=data  
 alphas=[0.01,0.02,0.05,0.1,0.2,0.5,1,2,5,10,20,50,100,200,500,1000]  
 scores=[]  
 for i,alpha in enumerate(alphas):  
 regr = linear\_model.Lasso(alpha=alpha)  
 regr.fit(X\_train, y\_train)  
 scores.append(regr.score(X\_test, y\_test))  
 ## 绘图  
 fig=plt.figure()  
 ax=fig.add\_subplot(1,1,1)  
 ax.plot(alphas,scores)  
 ax.set\_xlabel(r"$\alpha$")  
 ax.set\_ylabel(r"score")  
 ax.set\_xscale('log')  
 ax.set\_title("Lasso")  
 plt.show()  
if \_\_name\_\_=='\_\_main\_\_':  
 X\_train,X\_test,y\_train,y\_test=load\_data() # 产生用于回归问题的数据集  
 test\_Lasso(X\_train,X\_test,y\_train,y\_test) # 调用 test\_Lasso  
 # test\_Lasso\_alpha(X\_train,X\_test,y\_train,y\_test) # 调用 test\_Lasso\_alpha

## Elasticnet回归-结合岭回归和lasso回归

# -\*- coding: utf-8 -\*-  
"""  
 广义线性模型： ElasticNet  
"""  
import matplotlib.pyplot as plt  
import numpy as np  
from sklearn import datasets, linear\_model,cross\_validation  
  
def load\_data():  
 '''  
 加载用于回归问题的数据集  
 :return: 一个元组，用于回归问题。元组元素依次为：训练样本集、测试样本集、训练样本集对应的值、测试样本集对应的值  
 '''  
 diabetes = datasets.load\_diabetes()#使用 scikit-learn 自带的一个糖尿病病人的数据集  
 return cross\_validation.train\_test\_split(diabetes.data,diabetes.target,  
 test\_size=0.25,random\_state=0) # 拆分成训练集和测试集，测试集大小为原始数据集大小的 1/4  
  
def test\_ElasticNet(\*data):  
 '''  
 测试 ElasticNet 的用法  
 :param data: 可变参数。它是一个元组，这里要求其元素依次为：训练样本集、测试样本集、训练样本的值、测试样本的值  
 :return: None  
 '''  
 X\_train,X\_test,y\_train,y\_test=data  
 regr = linear\_model.ElasticNet()  
 regr.fit(X\_train, y\_train)  
 print('Coefficients:%s, intercept %.2f'%(regr.coef\_,regr.intercept\_))  
 print("Residual sum of squares: %.2f"% np.mean((regr.predict(X\_test) - y\_test) \*\* 2))  
 print('Score: %.2f' % regr.score(X\_test, y\_test))  
def test\_ElasticNet\_alpha\_rho(\*data):  
 '''  
 测试 ElasticNet 的预测性能随 alpha 和 l1\_ratio 的影响  
 :param data: 可变参数。它是一个元组，这里要求其元素依次为：训练样本集、测试样本集、训练样本的值、测试样本的值  
 :return: None  
 '''  
 X\_train,X\_test,y\_train,y\_test=data  
 alphas=np.logspace(-2,2)  
 rhos=np.linspace(0.01,1)  
 scores=[]  
 for alpha in alphas:  
 for rho in rhos:  
 regr = linear\_model.ElasticNet(alpha=alpha,l1\_ratio=rho)  
 regr.fit(X\_train, y\_train)  
 scores.append(regr.score(X\_test, y\_test))  
 ## 绘图  
 alphas, rhos = np.meshgrid(alphas, rhos)  
 scores=np.array(scores).reshape(alphas.shape)  
 from mpl\_toolkits.mplot3d import Axes3D  
 from matplotlib import cm  
 fig=plt.figure()  
 ax=Axes3D(fig)  
 surf = ax.plot\_surface(alphas, rhos, scores, rstride=1, cstride=1, cmap=cm.jet,  
 linewidth=0, antialiased=False)  
 fig.colorbar(surf, shrink=0.5, aspect=5)  
 ax.set\_xlabel(r"$\alpha$")  
 ax.set\_ylabel(r"$\rho$")  
 ax.set\_zlabel("score")  
 ax.set\_title("ElasticNet")  
 plt.show()  
if \_\_name\_\_=='\_\_main\_\_':  
 X\_train,X\_test,y\_train,y\_test=load\_data() # 产生用于回归问题的数据集  
 test\_ElasticNet(X\_train,X\_test,y\_train,y\_test) # 调用 test\_ElasticNet  
 # test\_ElasticNet\_alpha\_rho(X\_train,X\_test,y\_train,y\_test) # 调用 test\_ElasticNet\_alpha\_rho

## 逻辑斯特回归（分类）

# -\*- coding: utf-8 -\*-  
"""  
 广义线性模型：Logistic 回归（也称作对数几率回归）  
  
"""  
import matplotlib.pyplot as plt  
import numpy as np  
from sklearn import datasets, linear\_model,cross\_validation  
  
def load\_data():  
 '''  
 加载用于分类问题的数据集  
 :return: 一个元组，用于分类问题。元组元素依次为：训练样本集、测试样本集、训练样本集对应的标记、测试样本集对应的标记  
 '''  
 iris=datasets.load\_iris() # 使用 scikit-learn 自带的 iris 数据集  
 X\_train=iris.data  
 y\_train=iris.target  
 return cross\_validation.train\_test\_split(X\_train, y\_train,test\_size=0.25,  
 random\_state=0,stratify=y\_train)# 分层采样拆分成训练集和测试集，测试集大小为原始数据集大小的 1/4  
def test\_LogisticRegression(\*data):  
 '''  
 测试 LogisticRegression 的用法  
 :param data: 可变参数。它是一个元组，这里要求其元素依次为：训练样本集、测试样本集、训练样本的标记、测试样本的标记  
 :return: None  
 '''  
 X\_train,X\_test,y\_train,y\_test=data  
 regr = linear\_model.LogisticRegression()  
 regr.fit(X\_train, y\_train)  
 print('Coefficients:%s, intercept %s'%(regr.coef\_,regr.intercept\_))  
 print('Score: %.2f' % regr.score(X\_test, y\_test))  
def test\_LogisticRegression\_multinomial(\*data):  
 '''  
 测试 LogisticRegression 的预测性能随 multi\_class 参数的影响  
 :param data: 可变参数。它是一个元组，这里要求其元素依次为：训练样本集、测试样本集、训练样本的标记、测试样本的标记  
 :return: None  
 '''  
 X\_train,X\_test,y\_train,y\_test=data  
 regr = linear\_model.LogisticRegression(multi\_class='multinomial',solver='lbfgs')  
 regr.fit(X\_train, y\_train)  
 print('Coefficients:%s, intercept %s'%(regr.coef\_,regr.intercept\_))  
 print('Score: %.2f' % regr.score(X\_test, y\_test))  
def test\_LogisticRegression\_C(\*data):  
 '''  
 测试 LogisticRegression 的预测性能随 C 参数的影响  
 :param data: 可变参数。它是一个元组，这里要求其元素依次为：训练样本集、测试样本集、训练样本的标记、测试样本的标记  
 :return: None  
 '''  
 X\_train,X\_test,y\_train,y\_test=data  
 Cs=np.logspace(-2,4,num=100)  
 scores=[]  
 for C in Cs:  
 regr = linear\_model.LogisticRegression(C=C)  
 regr.fit(X\_train, y\_train)  
 scores.append(regr.score(X\_test, y\_test))  
 ## 绘图  
 fig=plt.figure()  
 ax=fig.add\_subplot(1,1,1)  
 ax.plot(Cs,scores)  
 ax.set\_xlabel(r"C")  
 ax.set\_ylabel(r"score")  
 ax.set\_xscale('log')  
 ax.set\_title("LogisticRegression")  
 plt.show()  
  
if \_\_name\_\_=='\_\_main\_\_':  
 X\_train,X\_test,y\_train,y\_test=load\_data() # 加载用于分类的数据集  
 test\_LogisticRegression(X\_train,X\_test,y\_train,y\_test) # 调用 test\_LogisticRegression  
 # test\_LogisticRegression\_multinomial(X\_train,X\_test,y\_train,y\_test) # 调用 test\_LogisticRegression\_multinomial  
 # test\_LogisticRegression\_C(X\_train,X\_test,y\_train,y\_test) # 调用 test\_LogisticRegression\_C

## 6.LDA线性判别模型

# -\*- coding: utf-8 -\*-  
"""  
 广义线性模型：线性判别分析LDA  
"""  
import matplotlib.pyplot as plt  
import numpy as np  
from sklearn import datasets, discriminant\_analysis,cross\_validation  
  
def load\_data():  
 '''  
 加载用于分类问题的数据集  
 :return: 一个元组，用于分类问题。元组元素依次为：训练样本集、测试样本集、训练样本集对应的标记、测试样本集对应的标记  
 '''  
 iris=datasets.load\_iris() # 使用 scikit-learn 自带的 iris 数据集  
 X\_train=iris.data  
 y\_train=iris.target  
 return cross\_validation.train\_test\_split(X\_train, y\_train,test\_size=0.25,  
 random\_state=0,stratify=y\_train)# 分层采样拆分成训练集和测试集，测试集大小为原始数据集大小的 1/4  
def test\_LinearDiscriminantAnalysis(\*data):  
 '''  
 测试 LinearDiscriminantAnalysis 的用法  
 :param data: 可变参数。它是一个元组，这里要求其元素依次为：训练样本集、测试样本集、训练样本的标记、测试样本的标记  
 :return: None  
 '''  
 X\_train,X\_test,y\_train,y\_test=data  
 lda = discriminant\_analysis.LinearDiscriminantAnalysis()  
 lda.fit(X\_train, y\_train)  
 print('Coefficients:%s, intercept %s'%(lda.coef\_,lda.intercept\_))  
 print('Score: %.2f' % lda.score(X\_test, y\_test))  
def plot\_LDA(converted\_X,y):  
 '''  
 绘制经过 LDA 转换后的数据  
 :param converted\_X: 经过 LDA转换后的样本集  
 :param y: 样本集的标记  
 :return: None  
 '''  
 from mpl\_toolkits.mplot3d import Axes3D  
 fig=plt.figure()  
 ax=Axes3D(fig)  
 colors='rgb'  
 markers='o\*s'  
 for target,color,marker in zip([0,1,2],colors,markers):  
 pos=(y==target).ravel()  
 X=converted\_X[pos,:]  
 ax.scatter(X[:,0], X[:,1], X[:,2],color=color,marker=marker,  
 label="Label %d"%target)  
 ax.legend(loc="best")  
 fig.suptitle("Iris After LDA")  
 plt.show()  
def run\_plot\_LDA():  
 '''  
 执行 plot\_LDA 。其中数据集来自于 load\_data() 函数  
 :return: None  
 '''  
 X\_train,X\_test,y\_train,y\_test=load\_data()  
 X=np.vstack((X\_train,X\_test))  
 Y=np.vstack((y\_train.reshape(y\_train.size,1),y\_test.reshape(y\_test.size,1)))  
 lda = discriminant\_analysis.LinearDiscriminantAnalysis()  
 lda.fit(X, Y)  
 converted\_X=np.dot(X,np.transpose(lda.coef\_))+lda.intercept\_  
 plot\_LDA(converted\_X,Y)  
def test\_LinearDiscriminantAnalysis\_solver(\*data):  
 '''  
 测试 LinearDiscriminantAnalysis 的预测性能随 solver 参数的影响  
 :param data: 可变参数。它是一个元组，这里要求其元素依次为：训练样本集、测试样本集、训练样本的标记、测试样本的标记  
 :return: None  
 '''  
 X\_train,X\_test,y\_train,y\_test=data  
 solvers=['svd','lsqr','eigen']  
 for solver in solvers:  
 if(solver=='svd'):  
 lda = discriminant\_analysis.LinearDiscriminantAnalysis(solver=solver)  
 else:  
 lda = discriminant\_analysis.LinearDiscriminantAnalysis(solver=solver,  
 shrinkage=None)  
 lda.fit(X\_train, y\_train)  
 print('Score at solver=%s: %.2f' %(solver, lda.score(X\_test, y\_test)))  
def test\_LinearDiscriminantAnalysis\_shrinkage(\*data):  
 '''  
 测试 LinearDiscriminantAnalysis 的预测性能随 shrinkage 参数的影响  
 :param data: 可变参数。它是一个元组，这里要求其元素依次为：训练样本集、测试样本集、训练样本的标记、测试样本的标记  
 :return: None  
 '''  
 X\_train,X\_test,y\_train,y\_test=data  
 shrinkages=np.linspace(0.0,1.0,num=20)  
 scores=[]  
 for shrinkage in shrinkages:  
 lda = discriminant\_analysis.LinearDiscriminantAnalysis(solver='lsqr',  
 shrinkage=shrinkage)  
 lda.fit(X\_train, y\_train)  
 scores.append(lda.score(X\_test, y\_test))  
 ## 绘图  
 fig=plt.figure()  
 ax=fig.add\_subplot(1,1,1)  
 ax.plot(shrinkages,scores)  
 ax.set\_xlabel(r"shrinkage")  
 ax.set\_ylabel(r"score")  
 ax.set\_ylim(0,1.05)  
 ax.set\_title("LinearDiscriminantAnalysis")  
 plt.show()  
  
if \_\_name\_\_=='\_\_main\_\_':  
 X\_train,X\_test,y\_train,y\_test=load\_data() # 产生用于分类的数据集  
 test\_LinearDiscriminantAnalysis(X\_train,X\_test,y\_train,y\_test) # 调用 test\_LinearDiscriminantAnalysis  
 # run\_plot\_LDA() # 调用 run\_plot\_LDA  
 # test\_LinearDiscriminantAnalysis\_solver(X\_train,X\_test,y\_train,y\_test) # 调用 test\_LinearDiscriminantAnalysis\_solver  
 # test\_LinearDiscriminantAnalysis\_shrinkage(X\_train,X\_test,y\_train,y\_test) # 调用 test\_LinearDiscriminantAnalysis\_shrinkage

# API详解

# **[sklearn.linear\_model](http://scikit-learn.org/stable/modules/classes.html" \l "module-sklearn.linear_model" \o "sklearn.linear_model).LinearRegression**

*class*sklearn.linear\_model.**LinearRegression**(*fit\_intercept=True*, *normalize=False*, *copy\_X=True*, *n\_jobs=1*)[[source]](https://github.com/scikit-learn/scikit-learn/blob/ef5cb84a/sklearn/linear_model/base.py" \l "L405)

Ordinary least squares Linear Regression.

|  |  |
| --- | --- |
| **Parameters:** | ****fit\_intercept**** : boolean, optional, default True一个布尔值，指定是否需要计算b的值，如果为False，那么不计算b值。  whether to calculate the intercept for this model. If set to False, no intercept will be used in calculations (e.g. data is expected to be already centered).  ****normalize**** : boolean, optional, default False一个布尔值，如果为true那么训练样本会在回归之前会被归一化。  This parameter is ignored when fit\_intercept is set to False. If True, the regressors X will be normalized before regression by subtracting the mean and dividing by the l2-norm. If you wish to standardize, please use **[sklearn.preprocessing.StandardScaler](http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html" \l "sklearn.preprocessing.StandardScaler" \o "sklearn.preprocessing.StandardScaler)** before calling fit on an estimator with normalize=False.  ****copy\_X**** : boolean, optional, default True 一个布尔值，如果为True，则会赋值X  If True, X will be copied; else, it may be overwritten.  ****n\_jobs**** : int, optional, default 1一个正数。任务并行时指定的CPU数量，如果为-1则使用所有可用的cpu.  The number of jobs to use for the computation. If -1 all CPUs are used. This will only provide speedup for n\_targets > 1 and sufficient large problems. |
| **Attributes:** | ****coef\_**** : array, shape (n\_features, ) or (n\_targets, n\_features)权重向量  Estimated coefficients for the linear regression problem. If multiple targets are passed during the fit (y 2D), this is a 2D array of shape (n\_targets, n\_features), while if only one target is passed, this is a 1D array of length n\_features.  ****intercept\_**** : array b的值  Independent term in the linear model. |
|  |  |

**Methods**

|  |  |
| --- | --- |
| **[fit](http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html" \l "sklearn.linear_model.LinearRegression.fit" \o "sklearn.linear_model.LinearRegression.fit)**(X, y[, sample\_weight]) | Fit linear model.训练线性回归模型 |
| **[get\_params](http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html" \l "sklearn.linear_model.LinearRegression.get_params" \o "sklearn.linear_model.LinearRegression.get_params)**([deep]) | Get parameters for this estimator. |
| **[predict](http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html" \l "sklearn.linear_model.LinearRegression.predict" \o "sklearn.linear_model.LinearRegression.predict)**(X) | Predict using the linear model用于模型的预测，返回预测值 |
| **[score](http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html" \l "sklearn.linear_model.LinearRegression.score" \o "sklearn.linear_model.LinearRegression.score)**(X, y[, sample\_weight]) | Returns the coefficient of determination R^2 of the prediction.  返回预测性能得分，设预测集为T，真实值为yi，真实值的均值为，则    Score不超过1，可能为负值，效果很差  Score越大，预测性能越好 |
| **[set\_params](http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html" \l "sklearn.linear_model.LinearRegression.set_params" \o "sklearn.linear_model.LinearRegression.set_params)**(\*\*params) | Set the parameters of this estimator. |

岭回归：

# **[sklearn.linear\_model](http://scikit-learn.org/stable/modules/classes.html" \l "module-sklearn.linear_model" \o "sklearn.linear_model).Ridge 岭回归**

*class*sklearn.linear\_model.**Ridge**(*alpha=1.0*, *fit\_intercept=True*, *normalize=False*, *copy\_X=True*, *max\_iter=None*, *tol=0.001*, *solver=’auto’*, *random\_state=None*)[[source]](https://github.com/scikit-learn/scikit-learn/blob/ef5cb84a/sklearn/linear_model/ridge.py" \l "L512)

Linear least squares with l2 regularization.**加入L2范数惩罚项的线性回归**

This model solves a regression model where the loss function is the linear least squares function and regularization is given by the l2-norm. Also known as Ridge Regression or Tikhonov regularization. This estimator has built-in support for multi-variate regression (i.e., when y is a 2d-array of shape [n\_samples, n\_targets]).

Read more in the [User Guide](http://scikit-learn.org/stable/modules/linear_model.html" \l "ridge-regression).

|  |  |
| --- | --- |
| **Parameters:** | ****alpha**** : {float, array-like}, shape (n\_targets)其值越大正则化项的占比越大  Regularization strength; must be a positive float. Regularization improves the conditioning of the problem and reduces the variance of the estimates. Larger values specify stronger regularization. Alpha corresponds to C^-1 in other linear models such as LogisticRegression or LinearSVC. If an array is passed, penalties are assumed to be specific to the targets. Hence they must correspond in number.  ****fit\_intercept**** : boolean一个布尔值，指定是否需要计算b的值。如果为false，那么不计算b值（模型会假设你的数据已经中心化了）  Whether to calculate the intercept for this model. If set to false, no intercept will be used in calculations (e.g. data is expected to be already centered).  ****normalize**** : boolean, optional, default False一个布尔值，如果为true，那么训练样本会在回归之前被归一化  This parameter is ignored when fit\_intercept is set to False. If True, the regressors X will be normalized before regression by subtracting the mean and dividing by the l2-norm. If you wish to standardize, please use **[sklearn.preprocessing.StandardScaler](http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html" \l "sklearn.preprocessing.StandardScaler" \o "sklearn.preprocessing.StandardScaler)** before calling fit on an estimator with normalize=False.  ****copy\_X**** : boolean, optional, default True布尔值，为true，则会复制X  If True, X will be copied; else, it may be overwritten.  ****max\_iter**** : int, optional整数，最大迭代次数  Maximum number of iterations for conjugate gradient solver. For ‘sparse\_cg’ and ‘lsqr’ solvers, the default value is determined by scipy.sparse.linalg. For ‘sag’ solver, the default value is 1000.  ****tol**** : float一个浮点数，指定判定迭代迭代收敛与否的阈值  Precision of the solution.  ****solver**** : {‘auto’, ‘svd’, ‘cholesky’, ‘lsqr’, ‘sparse\_cg’, ‘sag’, ‘saga’}  Solver to use in the computational routines:  指定求解最优解问题的算法，下列是可选值   * ‘auto’ chooses the solver automatically based on the type of data. * 根据数据集自动选择算法 * ‘svd’ uses a Singular Value Decomposition of X to compute the Ridge coefficients. More stable for singular matrices than ‘cholesky’. * 使用奇异值分解来计算回归系数 * ‘cholesky’ uses the standard scipy.linalg.solve function to obtain a closed-form solution. * 使用scipy.linalg.solve函数来求解 * ‘sparse\_cg’ uses the conjugate gradient solver as found in scipy.sparse.linalg.cg. As an iterative algorithm, this solver is more appropriate than ‘cholesky’ for large-scale data (possibility to set tol and max\_iter).使用scipy.sparse.linalg求解，速度最快 * ‘lsqr’ uses the dedicated regularized least-squares routine scipy.sparse.linalg.lsqr. It is the fastest but may not be available in old scipy versions. It also uses an iterative procedure. * ‘sag’ uses a Stochastic Average Gradient descent, and ‘saga’ uses its improved, unbiased version named SAGA. Both methods also use an iterative procedure, and are often faster than other solvers when both n\_samples and n\_features are large. Note that ‘sag’ and ‘saga’ fast convergence is only guaranteed on features with approximately the same scale. You can preprocess the data with a scaler from sklearn.preprocessing. * 使用随机平均梯度下降法求解最优化问题。   All last five solvers support both dense and sparse data. However, only ‘sag’ and ‘saga’ supports sparse input when fit\_intercept is True.所有最后五个求解器都支持密集和稀疏数据。 但是，当fit\_intercept为True时，只有'sag'和'saga'支持稀疏输入。  ***New in version 0.17****:*Stochastic Average Gradient descent solver.  *New in version 0.19:*SAGA solver.  ****random\_state**** : int, RandomState instance or None, optional, default None  The seed of the pseudo random number generator to use when shuffling the data. If int, random\_state is the seed used by the random number generator; If RandomState instance, random\_state is the random number generator; If None, the random number generator is the RandomState instance used by np.random. Used when solver == ‘sag’.  如果为整数，则它值定了随机数生成器的种子  如果为RandomState 实例，则制定可随机数生成器  如果为None，则使用默认随机数生成器  *New in version 0.17: random\_state* to support Stochastic Average Gradient. |
| **Attributes:** | ****coef\_**** :权重向量 array, shape (n\_features,) or (n\_targets, n\_features)  Weight vector(s).  ****intercept\_**** : b值--float | array, shape = (n\_targets,)  Independent term in decision function. Set to 0.0 if fit\_intercept = False.  ****n\_iter\_**** :实际迭代次数 array or None, shape (n\_targets,)  Actual number of iterations for each target. Available only for sag and lsqr solvers. Other solvers will return None.  *New in version 0.17.* |

**Methods**

|  |  |
| --- | --- |
| **[fit](http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Ridge.html" \l "sklearn.linear_model.Ridge.fit" \o "sklearn.linear_model.Ridge.fit)**(X, y[, sample\_weight]) | Fit Ridge regression model 训练岭回归线性模型 |
| **[get\_params](http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Ridge.html" \l "sklearn.linear_model.Ridge.get_params" \o "sklearn.linear_model.Ridge.get_params)**([deep]) | Get parameters for this estimator. |
| **[predict](http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Ridge.html" \l "sklearn.linear_model.Ridge.predict" \o "sklearn.linear_model.Ridge.predict)**(X) | Predict using the linear model用模型进行预测，返回预测值 |
| **[score](http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Ridge.html" \l "sklearn.linear_model.Ridge.score" \o "sklearn.linear_model.Ridge.score)**(X, y[, sample\_weight]) | Returns the coefficient of determination R^2 of the prediction.  返回预测性能得分，和线性回归的公式一样，值越大性能越好 |
| **[set\_params](http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Ridge.html" \l "sklearn.linear_model.Ridge.set_params" \o "sklearn.linear_model.Ridge.set_params)**(\*\*params) | Set the parameters of this estimator. |

Lasso回归

# **[sklearn.linear\_model](http://scikit-learn.org/stable/modules/classes.html" \l "module-sklearn.linear_model" \o "sklearn.linear_model).Lasso 回归**

*class*sklearn.linear\_model.**Lasso**(*alpha=1.0*, *fit\_intercept=True*, *normalize=False*, *precompute=False*, *copy\_X=True*, *max\_iter=1000*, *tol=0.0001*, *warm\_start=False*, *positive=False*, *random\_state=None*, *selection=’cyclic’*)[[source]](https://github.com/scikit-learn/scikit-learn/blob/ef5cb84a/sklearn/linear_model/coordinate_descent.py" \l "L797)

Linear Model trained with L1 prior as regularizer (aka the Lasso)

The optimization objective for Lasso is:

(1 / (2 \* n\_samples)) \* ||y - Xw||^2\_2 + alpha \* ||w||\_1

Technically the Lasso model is optimizing the same objective function as the Elastic Net with l1\_ratio=1.0 (no L2 penalty).

Read more in the [User Guide](http://scikit-learn.org/stable/modules/linear_model.html" \l "lasso).

|  |  |
| --- | --- |
| **Parameters:** | ****alpha**** : float, optional 值越大正则化项占比越大  Constant that multiplies the L1 term. Defaults to 1.0. alpha = 0 is equivalent to an ordinary least square, solved by the **[LinearRegression](http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html" \l "sklearn.linear_model.LinearRegression" \o "sklearn.linear_model.LinearRegression)** object. For numerical reasons, using alpha = 0 with the Lasso object is not advised. Given this, you should use the **[LinearRegression](http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html" \l "sklearn.linear_model.LinearRegression" \o "sklearn.linear_model.LinearRegression)** object.  ****fit\_intercept**** : boolean 一个布尔值，指定时候需要计算b值，如果weiFalse，那么不会计算b值（模型已经假设你已经中心化了）。  whether to calculate the intercept for this model. If set to false, no intercept will be used in calculations (e.g. data is expected to be already centered).  ****normalize**** : boolean, optional, default False如果为True，训练样本会在回归之前被归一化  This parameter is ignored when fit\_intercept is set to False. If True, the regressors X will be normalized before regression by subtracting the mean and dividing by the l2-norm. If you wish to standardize, please use **[sklearn.preprocessing.StandardScaler](http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html" \l "sklearn.preprocessing.StandardScaler" \o "sklearn.preprocessing.StandardScaler)** before calling fit on an estimator with normalize=False.  ****precompute**** : True | False | array-like, default=False一个布尔值或一个序列。他决定了是否提前计算Gram矩阵来加速运算。  Whether to use a precomputed Gram matrix to speed up calculations. If set to 'auto' let us decide. The Gram matrix can also be passed as argument. For sparse input this option is always True to preserve sparsity.  ****copy\_X**** : boolean, optional, default True 如果weiTrue，则会复制X  If True, X will be copied; else, it may be overwritten.  ****max\_iter**** : int, optional最大迭代次数  The maximum number of iterations  ****tol**** : float, optional指定判断迭代收敛与否的阈值  The tolerance for the optimization: if the updates are smaller than tol, the optimization code checks the dual gap for optimality and continues until it is smaller than tol.  ****warm\_start**** : bool, optional一个布尔值，如果为true，那么使用前一次训练结果继续训练。否则从头开始训练  When set to True, reuse the solution of the previous call to fit as initialization, otherwise, just erase the previous solution.  ****positive**** : bool, optional一个布尔值，如果为true，那么强制要求权重向量的分量都是正数。  When set to True, forces the coefficients to be positive.  ****random\_state**** : int, RandomState instance or None, optional, default None  一个整数或者一个Randonstate实例，或者为None  The seed of the pseudo random number generator that selects a random feature to update. If int, random\_state is the seed used by the random number generator; If RandomState instance, random\_state is the random number generator; If None, the random number generator is the RandomState instance used by np.random. Used when selection == ‘random’.  ****selection**** : str, default ‘cyclic’一个字符串，可以取值cyclic或random，指定了每次迭代的时候选择权重向量的那个分量来更新  Random：为整数，则它指定了随机选择权重向量的一个分量来更新  Cyclic：更新的时候，从前往后依次选择权重向量的一个分量来更新  If set to ‘random’, a random coefficient is updated every iteration rather than looping over features sequentially by default. This (setting to ‘random’) often leads to significantly faster convergence especially when tol is higher than 1e-4. |
| **Attributes:** | ****coef\_**** : 权重向量，array, shape (n\_features,) | (n\_targets, n\_features)  parameter vector (w in the cost function formula)  ****sparse\_coef\_**** : 稀疏系数scipy.sparse matrix, shape (n\_features, 1) | (n\_targets, n\_features)  sparse\_coef\_ is a readonly property derived from coef\_  ****intercept\_**** :b的值， float | array, shape (n\_targets,)  independent term in decision function.  ****n\_iter\_**** :实际迭代次数 int | array-like, shape (n\_targets,)  number of iterations run by the coordinate descent solver to reach the specified tolerance. |

**Methods**

|  |  |
| --- | --- |
| **[fit](http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Lasso.html" \l "sklearn.linear_model.Lasso.fit" \o "sklearn.linear_model.Lasso.fit)**(X, y[, check\_input]) | Fit model with coordinate descent.训练模型 |
| **[get\_params](http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Lasso.html" \l "sklearn.linear_model.Lasso.get_params" \o "sklearn.linear_model.Lasso.get_params)**([deep]) | Get parameters for this estimator. |
| **[path](http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Lasso.html" \l "sklearn.linear_model.Lasso.path" \o "sklearn.linear_model.Lasso.path)**(X, y[, l1\_ratio, eps, n\_alphas, …]) | Compute elastic net path with coordinate descent计算具有坐标下降的弹性网路径 |
| **[predict](http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Lasso.html" \l "sklearn.linear_model.Lasso.predict" \o "sklearn.linear_model.Lasso.predict)**(X) | Predict using the linear model 用模型进行预测 |
| **[score](http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Lasso.html" \l "sklearn.linear_model.Lasso.score" \o "sklearn.linear_model.Lasso.score)**(X, y[, sample\_weight]) | Returns the coefficient of determination R^2 of the prediction.返回预测性能得分 |
| **[set\_params](http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Lasso.html" \l "sklearn.linear_model.Lasso.set_params" \o "sklearn.linear_model.Lasso.set_params)**(\*\*params) | Set the parameters of this estimator. |

弹性网络回归

# **[sklearn.linear\_model](http://scikit-learn.org/stable/modules/classes.html" \l "module-sklearn.linear_model" \o "sklearn.linear_model).ElasticNet 回归**

*class*sklearn.linear\_model.**ElasticNet**(*alpha=1.0*, *l1\_ratio=0.5*, *fit\_intercept=True*, *normalize=False*, *precompute=False*, *max\_iter=1000*, *copy\_X=True*, *tol=0.0001*, *warm\_start=False*, *positive=False*, *random\_state=None*, *selection=’cyclic’*)[[source]](https://github.com/scikit-learn/scikit-learn/blob/ef5cb84a/sklearn/linear_model/coordinate_descent.py" \l "L510)

Linear regression with combined L1 and L2 priors as regularizer.

Minimizes the objective function:

1 / (2 \* n\_samples) \* ||y - Xw||^2\_2+ alpha \* l1\_ratio \* ||w||\_1+ 0.5 \* alpha \* (1 - l1\_ratio) \* ||w||^2\_2

If you are interested in controlling the L1 and L2 penalty separately, keep in mind that this is equivalent to:

a \* L1 + b \* L2

where:

alpha = a + b **and** l1\_ratio = a / (a + b)

The parameter l1\_ratio corresponds to alpha in the glmnet R package while alpha corresponds to the lambda parameter in glmnet. Specifically, l1\_ratio = 1 is the lasso penalty. Currently, l1\_ratio <= 0.01 is not reliable, unless you supply your own sequence of alpha.

Read more in the [User Guide](http://scikit-learn.org/stable/modules/linear_model.html" \l "elastic-net).

|  |  |
| --- | --- |
| **Parameters:** | ****alpha**** : float, optional**增加惩罚条件的常数。** 默认为1.0。请参阅这个参数的确切数学含义的说明.``alpha = 0``相当于一个普通的最小二乘法，由LinearRegression对象求解。 出于数值原因，不建议使用alpha = 0与Lasso对象。 给定这个，你应该使用LinearRegression对象。  Constant that multiplies the penalty terms. Defaults to 1.0. See the notes for the exact mathematical meaning of this parameter.``alpha = 0`` is equivalent to an ordinary least square, solved by the **[LinearRegression](http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html" \l "sklearn.linear_model.LinearRegression" \o "sklearn.linear_model.LinearRegression)** object. For numerical reasons, using alpha = 0 with the Lasso object is not advised. Given this, you should use the **[LinearRegression](http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html" \l "sklearn.linear_model.LinearRegression" \o "sklearn.linear_model.LinearRegression)** object.  ****l1\_ratio**** : float--ElasticNet混合参数，0 <= l1\_ratio <= 1。对于l1\_ratio = 0，惩罚是L2惩罚。 对于l1\_ratio = 1，它是一个L1惩罚。 对于0 <l1\_ratio <1，惩罚是L1和L2的组合。  The ElasticNet mixing parameter, with 0 <= l1\_ratio <= 1. For l1\_ratio = 0 the penalty is an L2 penalty. For l1\_ratio = 1 it is an L1 penalty. For 0 < l1\_ratio < 1, the penalty is a combination of L1 and L2.  ****fit\_intercept**** : bool是否应该估计截距b的值。 如果False，假定数据已经居中。  Whether the intercept should be estimated or not. If False, the data is assumed to be already centered.  ****normalize**** : boolean, optional, default False一个布尔值，如果为True那么训练样本会在回归之前被归一化。  This parameter is ignored when fit\_intercept is set to False. If True, the regressors X will be normalized before regression by subtracting the mean and dividing by the l2-norm. If you wish to standardize, please use **[sklearn.preprocessing.StandardScaler](http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html" \l "sklearn.preprocessing.StandardScaler" \o "sklearn.preprocessing.StandardScaler)** before calling fit on an estimator with normalize=False.  ****precompute**** : True | False | array-like 布尔值或序列，决定了是否提前计算Gram矩阵来加速计算  Whether to use a precomputed Gram matrix to speed up calculations. The Gram matrix can also be passed as argument. For sparse input this option is always True to preserve sparsity.  ****max\_iter**** : int, optional最大迭代次数  The maximum number of iterations  ****copy\_X**** : boolean, optional, default True 为True复制x  If True, X will be copied; else, it may be overwritten.  ****tol**** : float, optional指定判断迭代收敛与否的阈值  The tolerance for the optimization: if the updates are smaller than tol, the optimization code checks the dual gap for optimality and continues until it is smaller than tol.  ****warm\_start**** : bool, optional如果为true，那么使用前一次训练结果继续训练，否则从头开始训练  When set to True, reuse the solution of the previous call to fit as initialization, otherwise, just erase the previous solution.  ****positive**** : bool, optional 如果为True，那么强制要求权重向量的分量都为正数  When set to True, forces the coefficients to be positive.  ****random\_state**** : int, RandomState instance or None, optional, default None  The seed of the pseudo random number generator that selects a random feature to update. If int, random\_state is the seed used by the random number generator; If RandomState instance, random\_state is the random number generator; If None, the random number generator is the RandomState instance used by np.random. Used when selection == ‘random’.  ****selection**** : str, default ‘cyclic’一个字符串，取值为cyclic’或random  Random更新的时候，随机选择权重向量的一个分量过来更新  cyclic’更新的时候，从前往后依次选择权重向量的一个分量来更新  If set to ‘random’, a random coefficient is updated every iteration rather than looping over features sequentially by default. This (setting to ‘random’) often leads to significantly faster convergence especially when tol is higher than 1e-4. |
| **Attributes:** | ****coef\_**** : 权重向量array, shape (n\_features,) | (n\_targets, n\_features)  parameter vector (w in the cost function formula)  ****sparse\_coef\_**** : scipy.sparse matrix, shape (n\_features, 1) | (n\_targets, n\_features)  sparse\_coef\_ is a readonly property derived from coef\_  ****intercept\_**** :b值 float | array, shape (n\_targets,)  independent term in decision function.  ****n\_iter\_**** : 实际迭代次数array-like, shape (n\_targets,)  number of iterations run by the coordinate descent solver to reach the specified tolerance. |

**Methods**

|  |  |
| --- | --- |
| **[fit](http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.ElasticNet.html" \l "sklearn.linear_model.ElasticNet.fit" \o "sklearn.linear_model.ElasticNet.fit)**(X, y[, check\_input]) | Fit model with coordinate descent.训练模型 |
| **[get\_params](http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.ElasticNet.html" \l "sklearn.linear_model.ElasticNet.get_params" \o "sklearn.linear_model.ElasticNet.get_params)**([deep]) | Get parameters for this estimator. |
| **[path](http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.ElasticNet.html" \l "sklearn.linear_model.ElasticNet.path" \o "sklearn.linear_model.ElasticNet.path)**(X, y[, l1\_ratio, eps, n\_alphas, …]) | Compute elastic net path with coordinate descent |
| **[predict](http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.ElasticNet.html" \l "sklearn.linear_model.ElasticNet.predict" \o "sklearn.linear_model.ElasticNet.predict)**(X) | Predict using the linear model模型预测，返回预测值 |
| **[score](http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.ElasticNet.html" \l "sklearn.linear_model.ElasticNet.score" \o "sklearn.linear_model.ElasticNet.score)**(X, y[, sample\_weight]) | Returns the coefficient of determination R^2 of the prediction.返回预测性能得分 |
| **[set\_params](http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.ElasticNet.html" \l "sklearn.linear_model.ElasticNet.set_params" \o "sklearn.linear_model.ElasticNet.set_params)**(\*\*params) | Set the parameters of this estimator. |